

NOVEL HIGH-QUALITY VARIATIONAL IMAGE
DECOMPOSITION MODELS AND THEIR APPLICATIONS

REZA SHAHIDI





Library and Archives
Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-55384-8
Our file Notre référence
ISBN: 978-0-494-55384-8

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.


Canada

Novel High-Quality Variational Image Decomposition Models and their Applications

by

© Reza Shahidi

A thesis submitted to the
School of Graduate Studies
in partial fulfilment of the
requirements for the degree of
Doctor of Philosophy

Faculty of Engineering and Applied Science
Memorial University of Newfoundland

January 2008

St. John's

Newfoundland

Dedicated to my parents

Mahboobeh and Fereidoon

for their continuous and unconditional love and support
throughout my life.

Abstract

This thesis examines the problem of variational image decomposition, as first introduced and developed by Meyer in [1], and its applications to textured image discrimination and textured image denoising. Image decomposition generally refers to the splitting of an image f into the sum of two or more components, e.g. u , a cartoon component, and v , a texture component. After a brief overview of the use of partial differential equations in image processing, which has become very widespread in recent years, a novel image decomposition model called Improved Edge Segregation, based on the pioneering work of Vese and Osher [2], is put forward, which gives better decomposition results, as it better separates cartoon and texture edges into their proper components. Decomposition with Improved Edge Segregation is generally performed in less time than that from Vese and Osher's model, and gives superior (better quality and faster) texture discrimination results when used in conjunction with Active Contours without Edges [3]. Then a new model, called the Simultaneous Decomposition/Discrimination model, which simultaneously decomposes and segments a textured image is described, also improving decomposition and discrimination quality. Extensions to the image decomposition model of Osher, Solé and Vese [4], which is itself based on the H^{-1} -

norm, are subsequently put forward. One such extension decorrelates the cartoon and texture components by using an energy term based on the correlation coefficient between them in local windows, giving better decomposition results. The other combines the decomposition and nonlinear diffusion frameworks, for the purposes of ameliorating denoising performance. First, Perona and Malik [5] nonlinear diffusion is incorporated into decomposition, and subsequently the framework of image denoising with Oriented Laplacians is incorporated. Finally, two models where texture is represented by one subcomponent are presented, one which requires precomputation of image orientations, Orientation-Adaptive Decomposition, and another, Eikonal Orientation-Adaptive Decomposition, which does not. Orientation-Adaptive Decomposition is applied to both the decomposition and the denoising of oriented textures, and some theoretical discussion of Eikonal Orientation-Adaptive Decomposition included. Finally some conclusions and suggestions for future work are given based on the research presented in the thesis.

Acknowledgements

I would like to thank my supervisor, Dr. Cecilia Moloney for her guidance and for the freedom that she gave me for my research, especially during the post-comprehensive phase of my doctoral programme. I appreciate the work of my supervisory committee consisting of Dr. Moloney, Dr. Eric Gill and Dr. Dennis Peters, in reviewing this thesis and suggesting changes to make it more readable. I would also like to acknowledge the financial support of NSERC for the Postgraduate Doctoral Prize that helped support me during my Ph.D. studies. I am grateful for the financial support, for among other things, attending international conferences, of the Faculty of Engineering and Applied Science, the School of Graduate Studies and the Graduate Students' Union at the Memorial University of Newfoundland. Dr. Mahmoud Haddara must be thanked for being a mentor and somebody I could consult with no matter when it was needed.

Finally, the company of the many friends and acquaintances I made during my studies, both inside and outside of the Computer Engineering Research Laboratories (CERL), where I did much of my work, is appreciated, for making such a sizeable undertaking easier to see to fruition.

Contents

List of Symbols	xiv
List of Abbreviations	xvii
List of Tables	xix
List of Figures	xxi
1 Introduction	1
1.1 Motivation	1
1.2 Existing Image Processing/Analysis Challenges	1
1.2.1 Image Segmentation	2
1.2.2 Image Denoising	3
1.2.3 Image Compression and Transmission	5
1.2.4 Unification of the Solution of Image Processing Problems . . .	6
1.3 Introduction to Image Decomposition	7

1.3.1	Example of Image Decomposition	8
1.4	Representation of Digital Images with Image Decomposition	9
1.5	How Decomposition is Applied - Sequential and Simultaneous So- lutions	12
1.5.1	Simultaneous Solution of Image Processing Problems	12
1.5.2	Sequential and Simultaneous Solutions in this Thesis	13
1.6	Problem Statement	13
1.7	Proposed Method of Solution	16
1.8	A Roadmap and Original Contributions of this Thesis	16
2	A Review of Variational Image Processing Models and Other Necessary Tools	19
2.1	Introduction	19
2.2	Background: Nonlinear PDE Models in Image Processing	20
2.2.1	Example of the Energy Method in PDE-based Image Process- ing	22
2.2.1.1	The Energy Method	22
2.2.1.2	Example of the Energy Method - the Heat Equation	26
2.2.1.3	Perona-Malik Inhomogeneous Diffusion	31
2.2.1.4	Summary	34
2.3	Additive Operator Splitting	35

2.4	Image Decomposition Models - Mathematical Underpinnings	38
2.4.1	Background Image Decomposition Theory	38
2.4.2	Vese-Osher Decomposition	42
2.4.3	Osher-Solé-Vese Decomposition	45
2.5	Level Set Methods	47
2.6	Active Contours without Edges	50
2.7	Orientation and Coherence Calculation	54
2.7.1	Linear and Nonlinear Structure Tensors	54
2.7.2	Orientation Coherence	56
2.8	Chapter Summary	57
3	Improved Edge Segregation	59
3.1	Introduction	59
3.2	Example of Instability	60
3.3	Edge Segregation	62
3.3.1	Justification of Approximation	63
3.3.2	Euler-Lagrange Equations	65
3.3.3	Residual with Edge Segregation	66
3.4	Geometric Constraint	67
3.4.1	Explanation of requirement to blur g_1 and g_2	69
3.4.2	More on the Geometric Constraint Energy	69

3.5	Improved Edge Segregation	71
3.5.1	Proposed New Terms	71
3.5.2	Example of Nonlinear Diffusion Flow	73
3.5.3	Description of Parameters	74
3.5.4	Euler-Lagrange Equations	75
3.6	Parallel Implementation	77
3.7	Results	80
3.7.1	Test Set	80
3.7.2	Jacobi vs. Gauss-Seidel Iterations	82
3.7.3	Experimental Approach	86
3.7.4	Choice of Parameters	89
3.7.5	Decomposition Quality Measures	91
3.7.6	Serial Decomposition Results and Discussion	94
3.7.6.1	Decomposition Quality Measure Results	103
3.7.7	Parallel Decomposition Results and Discussion	104
3.7.8	Texture Discrimination	106
3.7.8.1	Background	106
3.7.8.2	Results	108
3.8	Conclusions	109
4	Simultaneous Decomposition and Discrimination	111

4.1	Introduction	111
4.2	Proposed Energy Functional	113
4.2.1	Meaning of New Terms	119
4.2.2	Calculation of Scale	121
4.2.3	Calculation of Coefficients based on Scale	123
4.3	Contrast Stretching	123
4.4	Methodology	125
4.4.1	Numerical Implementation	125
4.4.2	Varying the coefficients specific to the algorithm	128
4.4.3	Test Images	129
4.4.4	Parameter Selection	130
4.5	Results and Discussion	131
4.5.1	Two-Region Test Images	131
4.5.2	Test Image with More Than Two Regions	135
4.6	Conclusions	138
5	Modifying the Osher-Solé-Vese Decomposition Model	139
5.1	Introduction	139
5.2	Decorrelating decomposition components	140
5.2.1	Correlation coefficient	141
5.2.2	Simplifying Assumptions	143

5.2.3	Euler-Lagrange Equation	145
5.2.4	Test Images	146
5.2.5	Experimental Setup	148
5.2.6	DOSV Experimental Results	148
5.2.6.1	Qualitative Results	148
5.2.6.2	Quantitative Results	153
5.3	Incorporating Nonlinear Diffusion into the OSV Model	154
5.3.1	Perona-Malik Nonlinear Diffusion	154
5.3.2	Results for PMOSV	157
5.3.3	Diffusion with Oriented Laplacians	162
5.3.4	Determining Oriented and Non-Oriented Regions	164
5.3.5	The Oriented-Laplacian Osher-Solé-Vese Image Decomposition Model	165
5.3.6	Implementation Details	169
5.3.7	Results for OLOSV	171
5.4	Conclusions	176
6	Orientation-Adaptive Decomposition	181
6.1	Introduction	181
6.1.1	Motivation for the Simplification of the Vese-Osher Decomposition Model for Oriented Images	182

6.2	Orientation-Adaptive Decomposition Model	184
6.2.1	Derivation	184
6.2.2	Solution of Proposed Model	187
6.3	Division of Orientations into Sectors	188
6.3.1	Rotation of Derivative Filters	189
6.3.1.1	Rotated Derivative Filters for Decomposition	189
6.3.1.2	Rotated Derivative Filters for Denoising	193
6.4	Boundary Conditions	196
6.4.1	Boundary Conditions between Orientation Sectors	197
6.4.1.1	Oriented Gaussian and Linear Filters	197
6.4.1.2	Calculation of Isophote Angle of Linear Angular Filters	200
6.5	Phase Unwrapping	201
6.6	Generation of Texture from g Subcomponent	203
6.7	Non-Oriented Regions and How to Deal with Them	207
6.8	Eikonal Orientation-Adaptive Decomposition	209
6.8.1	Derivation of EOAD model	209
6.8.2	Numerical Solution of Euler-Lagrange Equations and Imple- mentation Details	211
6.8.3	Discussion of EOAD model	212

6.9	Experimental Setup and Results	214
6.9.1	Test Images	214
6.9.2	Experimental Setup	216
6.9.3	Results	218
6.9.3.1	OAD Results for Decomposition	218
6.9.3.2	OAD Denoising Results	222
6.10	Conclusions	235
7	Conclusions	237
7.1	Thesis Summary	237
7.2	Future Work	240
7.3	Final Reflections	244
A	Analytical Solution of OSV Decomposition Model	245
A.1	Introducing Subcomponents into the OSV Decomposition Model . .	245
B	Difference of Convex Functionals Solution of IES	249
B.1	Difference of Convex Functionals Solution	249
B.1.1	Some Elementary Convex Analysis	249
B.1.2	Difference of Convex Functionals Solution of IES	251
	Bibliography	255

List of Symbols

H^{-1}	Sobolev space modeling texture in OSV decomposition ...	iv
Ω	Rectangular image domain	2
σ	Standard deviation of noise or of Gaussian filter	4
f	The original image being decomposed	8
u	The cartoon (piecewise smooth) component output from decomposition of f	8
v	The texture component output from decomposition of f ..	8
r	The residual component ($f - u - v$) obtained from decomposition of f	8
$E(\cdot, \cdot, \dots)$	Energy functional with list of arguments equal to those functions over which the functional is taken	23
λ	Fidelity parameter for various decomposition models	24
Δt	Timestep for explicit/semi-implicit solution of PDE	25
g_1, g_2	Texture subcomponents in Meyer's model from which the texture component is formed	41
$\ \cdot\ _*$	Meyer's *-norm modeling texture	41

L^p	The space of p -integrable functions h , i.e. those h for which the L^p -norm $(\int_{\Omega}(h(x,y))^p dx dy)^{\frac{1}{p}}$ is finite	42
$ \cdot _p$	The L^p -norm of the argument function	43
ϕ	Level Set Function used by ACWE	48
δ_{ϵ}	Dirac delta function with regularization constant ϵ	53
H_{ϵ}	Heaviside (step) function with regularization constant ϵ . .	53
J	Structure Tensor (linear or nonlinear) for determination of gradient/isophote angles	54
$*$	Convolution Operator	55
$\theta_{i,j}$	Gradient Orientation measured from Structure Tensor . . .	56
G_{σ}	A Gaussian filter of mean 0 and standard deviation σ	62
f_{NLD}	Output of Non-linear Diffusion on f in preprocessing from IES decomposition	72
gmt	Gradient magnitude threshold for determination of edges used in calculation of decomposition quality measures . . .	92
$gmratio$	Threshold of ratio of $ \nabla f_{TV} $ to $ \nabla f $ to determine which edges are cartoon edges and which are texture edges for calculation of decomposition quality measures	92
CQ	Cartoon Quality Measure	93
TQ	Texture Quality Measure	93
CQ'	Cartoon Quality Measure over set of cartoon edges not close to texture edges in location	93
ϕ_1, ϕ_2	Level Set Functions used by Multiphase ACWE	112
f_{TV}	Output of TV flow on original image f	116

$Cov(X, Y)$	The covariance of the two random variables or functions X and Y	141
σ_X	The standard deviation of the random variable or function X	141
\bar{X}	The mean of the random variable or function X	141
$K(u)$	The curvature of the level lines of the function u	158
η	Gradient angle determined from Structure Tensor	163
ζ	Isophote angle determined from Structure Tensor	163
Ω_O	Image Region with Coherent Gradient Angles	166
Ω_{NO}	Image Region with non-Coherent Gradient Angles ($\Omega_O \cup \Omega_{NO} = \Omega$)	166
D_x, D_y	First-order Derivative Filters	169
θ	Gradient Orientation or Angle subcomponent from proposed Orientation-Adaptive Decomposition model	184
g	Texture subcomponent from Orientation-Adaptive and Eikonal Orientation-Adaptive Decomposition models	186
D_{xx}, D_{xy} and D_{yy}	Second-order Derivative Filters used in OAD	190
$\{P_i\}_{i=0}^3, P_{0,a},$ $P_{0,b}, P_{3,a}, P_{3,b}$	The Various Orientation Sectors with Different Rotated Derivative Filters in OAD	191
A^H	The matrix A flipped horizontally	196
∂H	The Subdifferential of the Energy Functional H	250
H^*	The Conjugate of the Functional H	252

List of Abbreviations

YCbCr	Color space representing colors with one luminance and two chroma components	10
PDE	Partial differential equation	16
IES decomposition	Improved Edge Segregation decomposition . . .	17
SDD	Simultaneous Decomposition/Discrimination .	17
DOSV decomposition	Decorrelated Osher-Solé-Vese decomposition . .	17
PMOSV decomposition	Perona-Malik-Osher-Solé-Vese decomposition .	17
OLOS decomposition	Oriented-Laplacian Osher-Solé-Vese decomposition	18
OAD	Orientation-Adaptive Decomposition	18
EOAD	Eikonal Orientation-Adaptive Decomposition .	18
AOS	Additive Operator Splitting	35
BV	Bounded Variation	39
TV	Total Variation	40
OSV decomposition	Osher-Solé-Vese decomposition	46
ACWE	Active Contours without Edges	50

V-O decomposition	Vese-Osher decomposition	57
MPI	Message Passing Interface	77
PNG	Portable Network Graphics	77
OSV2 decomposition	Variant of OSV decomposition where the fidelity parameter λ is computed separately in Ω_O and Ω_{NO}	171
SAR	Synthetic Aperture Radar	201
SNAPHU	Statistical-Cost, Network-Flow Algorithm for Phase Unwrapping - a piece of software for phase unwrapping used in OAD	202
OAD1	Version of OAD with Derivative Filter Rotation and Rotation-Invariant Derivative Filters	222
OAD2	Version of OAD without Derivative Filter Rotation and only with Rotation-Invariant Derivative Filters	222
OAD3	Version of OAD with Derivative Filter Rotation and mostly without Rotation-Invariant Derivative Filters	222

List of Tables

2.1	List of Tools/Models and the Decomposition Models Proposed in this Thesis which Use them	58
3.1	Parameter choices for IES on test images	90
3.2	Decomposition Algorithm, parameter choices, number of iterations required, and timing results for ACWE Texture Discrimination	90
3.3	Parameter choices for OSV decomposition	90
3.4	Number of Iterations and CPU Time required for Vese-Osher vs. proposed algorithms on Test Set	94
3.5	Decomposition Quality Measures of Test Images for V-O and IES Decomposition Models	103
3.6	CPU Time Required for IES Decomposition on Parallel Beowulf Cluster	105
4.1	Parameter choices for SDD on test images	130
4.2	Parameter choices for SDD on test images (cont.)	130

4.3	Number of iterations required for SDD, and V-O decomposition sequentially followed by ACWE discrimination on two-region test images	133
4.4	Time (secs) required for SDD and V-O decomposition sequentially followed by ACWE discrimination on two and three-region test images	134
5.1	Decomposition Quality at Cartoon and Texture Edges for OSV and proposed DOSV Decomposition Models	153
5.2	SNRs for OSV denoising vs. PMOSV denoising on test images	161
5.3	Time required for OSV denoising vs. PMOSV denoising on test images	162
5.4	SNRs for OSV denoising vs. PMOSV, OSV2 and OLOSV denoising on test images	171
5.5	Time and Number of Iterations required for OSV denoising vs. OSV2 and OLOSV denoising on test images	171
6.1	First-Order Derivative Filters for each Angular Sector $\{P_i\}_{i=0}^3$	192
6.2	Second-Order Derivative Filters for each Angular Sector $\{P_i\}_{i=0}^3$. . .	192
6.3	Signal-to-Noise Ratio (SNR) of OAD denoising vs. V-O and OSV denoising results	229
6.4	Execution Time and Number of Iterations Required for OAD Denoising vs. V-O and OSV Denoising	233
B.1	Terms in IES functional contained in each convex functional in DC functional	253

List of Figures

1.1	Example of Image Segmentation (from [6])	2
1.2	Aerial Image of Downtown Toronto (from [14]) with and without Noise	4
1.3	Segmentation of Mars Image into Sections that Can Be Compressed at Different Rates (from [16])	6
1.4	Sample Image Decomposition of zebra image	8
1.5	Flow Diagram of Sequential and Simultaneous Solutions of Image Decomposition and Another Image Processing Problem (Segmenta- tion)	14
2.1	(a) Original Lena image and (b) Lena image as a surface	21
2.2	Example of Isotropic Linear Diffusion	30
2.3	Example of Inhomogeneous Perona-Malik Diffusion	33
2.4	Formation of zero level-set by taking level cross-section of a 3-D sur- face	49

2.5	Sample Initialization of Level Set Function ϕ overlaid on $ g_1 $ sub-component of Image with Two Separate Textures	52
2.6	$H_{0.02}(z)$ and $\delta_{0.02}(z)$	53
3.1	Original barbara image and u component using Jacobi and Gauss-Seidel iteration implementations of the Vese-Osher model for image decomposition of barbara (50 iterations, $\mu = 0.05, \lambda = 0.05$)	60
3.2	Comparison of residual r component between V-O and Edge Segregation Methods	67
3.3	Artificial Image Violating the Geometric Constraint of Section 3.4	71
3.4	Example of AOS Total Variation Flow	74
3.5	1D Block Partitioning of Image	78
3.6	Diagram of MPI Communication Structure for IES	79
3.7	Images used to test image decomposition: (a) barbara, (b) barbzooom, (c) mosaic, (d) woodangle and (e) cactus	81
3.8	Illustration of Red-Black Ordering for Gauss-Seidel Iterations, taken from [67]	83
3.9	5-Colouring of Nodes for V-O Decomposition	84
3.10	Progression of Total Variation of u and L^2 -Norms of v and r vs. Iteration Number for V-O Model on mosaic Test Image	86
3.11	Detected Cartoon and Texture Edges in Test Image barbara Used for Calculation of Decomposition Quality	95
3.12	V-O Decomposition [2] ($\mu = \lambda = 0.05$) of barbara after 50 iterations	96

3.13 IES Decomposition of barbara ($\mu = \lambda = 0.05, \hat{\gamma}_1 = 0.12, \hat{\gamma}_2 = 1.2,$ $\sigma = 1$) after 43 iterations	97
3.14 V-O Decomposition [2] (u, v , and r) of mosaic ($\mu = 0.05, \lambda = 0.02$) after 101 iterations	99
3.15 IES Decomposition of mosaic ($\mu = 0.05, \lambda = 0.02, \hat{\gamma}_1 = 0.11, \hat{\gamma}_2 =$ $0.75, \sigma = 2$) after 99 iterations	100
3.16 V-O Decomposition [2] (g_1 and g_2) of mosaic ($\mu = 0.05, \lambda = 0.02$) after 101 iterations	101
3.17 V-O Decomposition [2] of cactus ($\mu = 0.05, \lambda = 0.05$) after 35 itera- tions	101
3.18 IES Decomposition of cactus ($\mu = 0.05, \lambda = 0.05, \hat{\gamma}_1 = 0.12, \hat{\gamma}_2 =$ $0.6, \sigma = 2$) after 44 iterations	102
3.19 OSV Decomposition of cactus after 100 iterations	102
3.20 Snapshot of Jumpshot Window running with Logfile from barbara on cluster	107
3.21 (a) ACWE Segmentation result for woodangle overlaid on $ g_1 $ ob- tained via IES decomposition, and (b) Segmentation result overlaid on original woodangle image for IES decomp.	109

3.22	(a) ACWE Segmentation result for mosaic overlaid on $ g_1 $ obtained via V-O decomposition, (b) Segmentation result overlaid on original mosaic image for V-O decomp., (c) ACWE Seg. result for mosaic overlaid on $ g_1 $ obtained via IES decomposition, and (d) Segmentation result overlaid on original mosaic image for IES decomp.	110
4.1	Iteration-Dependent Change in Contrast-Stretching Factor CS_g	125
4.2	Flow Diagram for SDD Algorithm	126
4.3	Test Images used in Experiments	129
4.4	Image decomposition results from V-O and SDD decomposition for mosaic1	131
4.5	Texture discrimination results from V-O and SDD decomposition for mosaic1	132
4.6	Texture discrimination results from V-O and SDD decomposition for mosaic2	133
4.7	Image decomposition results from V-O and SDD decomposition for mosaic2	134
4.8	Decomposition Result of Proposed SDD Algorithm for mosaic3 image	137
4.9	Discrimination Result of Proposed SDD Algorithm on mosaic3	137
5.1	Test Images used in this Chapter	147
5.2	Cartoon components of Decomposition of barbara with OSV and proposed DOSV models	150

5.3	Texture components of Decomposition of smallbarbara with OSV and proposed DOSV models	151
5.4	Texture components of Decomposition of lena with OSV and proposed DOSV models	152
5.5	Original and noisy test image	158
5.6	Denoising results from OSV Decomposition model	159
5.7	Denoising results from proposed PMOSV Decomposition model . . .	160
5.8	Illustration of Isophote and Gradient Directions	163
5.9	Cartoon (denoised) components from OSV, OSV2 and OLOSV Decompositions of barbara image	174
5.10	Noise components from OSV, OSV2 and OLOSV Decompositions of barbara image	175
5.11	Zoom on Noise Components from OSV, OSV2 and OLOSV decompositions of barbara image	176
5.12	Cartoon (denoised) components from OSV, OSV2 and OLOSV Decompositions of grass image	177
5.13	Noise components resulting from OSV, OSV2 and OLOSV Decompositions of grass image	178
5.14	Zoom on Noise Components from OSV, OSV2 and OLOSV Decompositions of grass image	179
6.1	Example of isophote and gradient directions (θ^\perp and θ respectively) .	184

6.2	Regions P_i with different derivative filters for decomposition (not denoising)	192
6.3	Regions P_i with different derivative filters for denoising	193
6.4	fingerprint test image	202
6.5	Example of Phase Unwrapping	203
6.6	Generation of Texture Component from g at Phase Discontinuities . .	204
6.7	Test Images used in Experiments	215
6.8	Comparison of Texture Components of Decomposition from V-O Algorithm and OAD <i>with</i> and <i>without</i> Derivative Filter Rotation	219
6.9	Comparison of Decomposition Results (without added input noise) from V-O and OAD on fingerprint with Zero Initial Condition for Texture Components	221
6.10	The Components of the OAD Decomposition of smallconcentric and Final Denoised Result <i>with</i> Filter Rotation and <i>with</i> Rotation-Invariant Derivative Filters (10 iterations, OAD1)	223
6.11	The Components of the OAD Decomposition of smallconcentric and Final Denoised Result <i>without</i> Filter Rotation and <i>without</i> Rotation-Invariant Derivative Filters (10 iterations, OAD2)	224
6.12	The Components of the OAD Decomposition of smallconcentric and Final Denoised Result <i>with</i> Filter Rotation and <i>without</i> Rotation-Invariant Filters (12 Iterations, OAD3)	226

6.13	Denoising Results on <code>smallconcentric</code> based on OSV and V-O decomposition models	227
6.14	Various components from V-O decomposition for denoising of noisy fingerprint	230
6.15	Various components from proposed Orientation-Adaptive Decomposition (OAD2) of noisy fingerprint	233
6.16	Various components from proposed Orientation-Adaptive Decomposition (OAD3, 9 iterations) of noisy fingerprint	234
6.17	Denoising results with Orientation-Adaptive Decomposition (OAD3, 6 iterations) on noisy eye	234
6.18	Denoising results with Orientation-Adaptive Decomposition (OAD3, 5 iterations) and Vese-Osher decomposition on noisy <code>barbzoom</code>	235
A.1	Osher-Solé-Vese decomposition of <code>barbara</code> with analytical g_1 and g_2 components after 314 iterations	248
B.1	Example of convex function h	250

CHAPTER 1

Introduction

1.1 Motivation

As the field of image processing grows more mature, along with its applications, new and challenging problems are encountered which require novel methods of solution. It will be seen that image decomposition [1] is a natural choice to solve these problems. However, some issues in turn exist with current image decomposition algorithms, namely poor quality and efficiency. This motivates the generation of new decomposition models, which perform better than the current state-of-the-art, from which follows the work in this thesis.

1.2 Existing Image Processing/Analysis Challenges

In the present day, digital images are becoming more and more ubiquitous, from cellular telephone cameras to barcode scanners in supermarkets. In most of these



Figure 1.1: Example of Image Segmentation (from [6])

instances, very fast (usually real-time) and accurate image processing and analysis algorithms are necessary in order to use these images to do something useful, such as sending an image from a camera half-way across the world, or checking out a carton of milk at the local grocery store.

Image segmentation, denoising and compression/transmission are examples of image processing problems for which solutions are needed very often in the real-world. These problems are now defined and described. Then in the following section, the common framework of image decomposition is described which facilitates the solution of all of these problems.

1.2.1 Image Segmentation

Image segmentation refers to partitioning an image domain Ω into disjoint regions, where each region is "uniform" in some sense, with respect to some characteristic. To define the disjointness of the regions more rigorously, if the (non-empty) regions of the segmentation are $\{R_i\}_{i=1}^n$, then $\bigcup_{i=1}^n R_i = \Omega$ and $R_i \cap R_j = \emptyset$ if $i \neq j$. The human visual system has been shown to do this top-down segmentation on its

own, separating images received by the optic nerve spontaneously and naturally to guide human actions in every sphere of life, from simple walking to parsing shapes into parts which are more easily processed [7].

For textured images, each region of the segmentation should consist of a uniform texture. An example of a segmented image, taken from [8], is shown in Figure 1.1. It is a result from the segmentation algorithm found in [9]. This result could actually be considered to be an oversegmentation, since the lizard would normally be considered in everyday life to represent one region, corresponding to that reptile.

Uniformity in the context of texture means that statistical properties of the texture are uniform throughout the region. There are many classical statistical approaches of measuring texture characteristics, e.g. spatial gray tone cooccurrence probabilities and measures based on the textures in the frequency domain [10].

Now that the problem of image segmentation has been discussed, the next image processing problem to be reviewed is the important problem of image denoising.

1.2.2 Image Denoising

In its usual sense, noise in an image can be defined to be stochastic variation in the intensities of image pixels not caused by objects or detail in the scene being imaged itself [11]. The noise, being stochastic in nature, can have a multitude of different probability distributions. In general, two common types of noise found in digital images are Gaussian and impulse noise. Additive white Gaussian noise is usually introduced during image acquisition, while impulse noise is principally introduced by transmission errors [12]. In this thesis, only additive white Gaussian

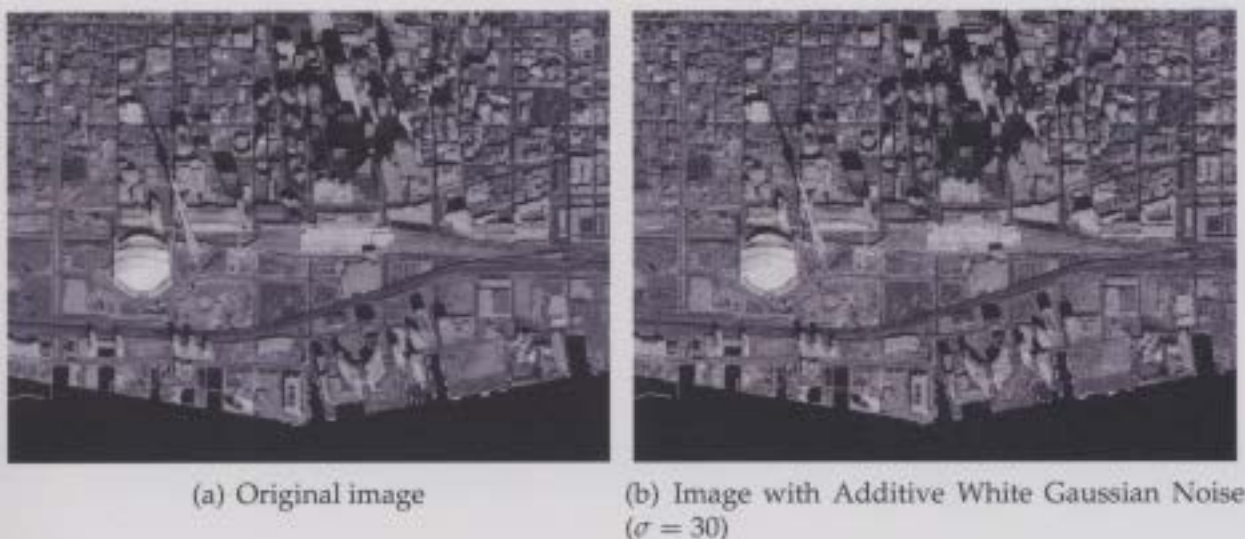


Figure 1.2: Aerial Image of Downtown Toronto (from [14]) with and without Noise

noise (AWGN) is considered, which has two parameters, the mean μ and standard deviation σ , however the research in this thesis could be extended to impulse noise models by using the methods in e.g. [13]. In this research, the assumption is always made that the mean of the additive white Gaussian noise is $\mu = 0$.

An example of a greyscale image (a one-meter resolution image of downtown Toronto, Canada by the IKONOS satellite on March 18, 2000 [14]) without any noise, and then corrupted with Additive White Gaussian Noise of standard deviation $\sigma = 30$ is shown in Figure 1.2. This image was selected since it contains many small-scale details, such as some buildings and roads, of which the obscurement by noise is more dramatic, than would be the case with larger scale structures.

The sources of image noise are diverse, however some include electronic noise, which comes from random electrical currents in the electronic components of the digital image/video system, and grain and structure noise, which appears because of the grain structure of the film in the non-digital case [15]. Another unavoidable

type of noise in digital images is quantization noise, caused by the inevitability of rounding when digital images are processed by computer, since pixel intensities must be represented by the computer's finite precision arithmetic. The challenge in image denoising is to remove the high-frequency noise present in the image without destroying the high-frequency edges. Simple linear filtering will not work in this regard because it cannot distinguish between high frequencies originating from noise or image edges. As described in Chapter 2, the anisotropic diffusion of Perona and Malik [5] is a variational method which has much better performance than pure linear filtering for the task of edge-preserving image denoising.

1.2.3 Image Compression and Transmission

Another problem faced in image processing in the digital world is the ever increasing amounts of image data. In order to store and transmit these images, it is desirable, and sometimes necessary to reduce the number of bits in the representation of the image. This problem could be related back to the problem of image segmentation discussed earlier, since often if a region in a segmentation is uniform in some sense, uniformity of the region can be taken advantage of to compress the data associated with it.

For instance, real-time texture segmentation is useful for segmenting photos of Mars, see Figure 1.3 from the Pathfinder mission [16]. The segmentation can be used to compress the images to be transmitted by the land rover, by compressing the less important sky at a higher rate than the rocks and sand in the image, which are more salient features.

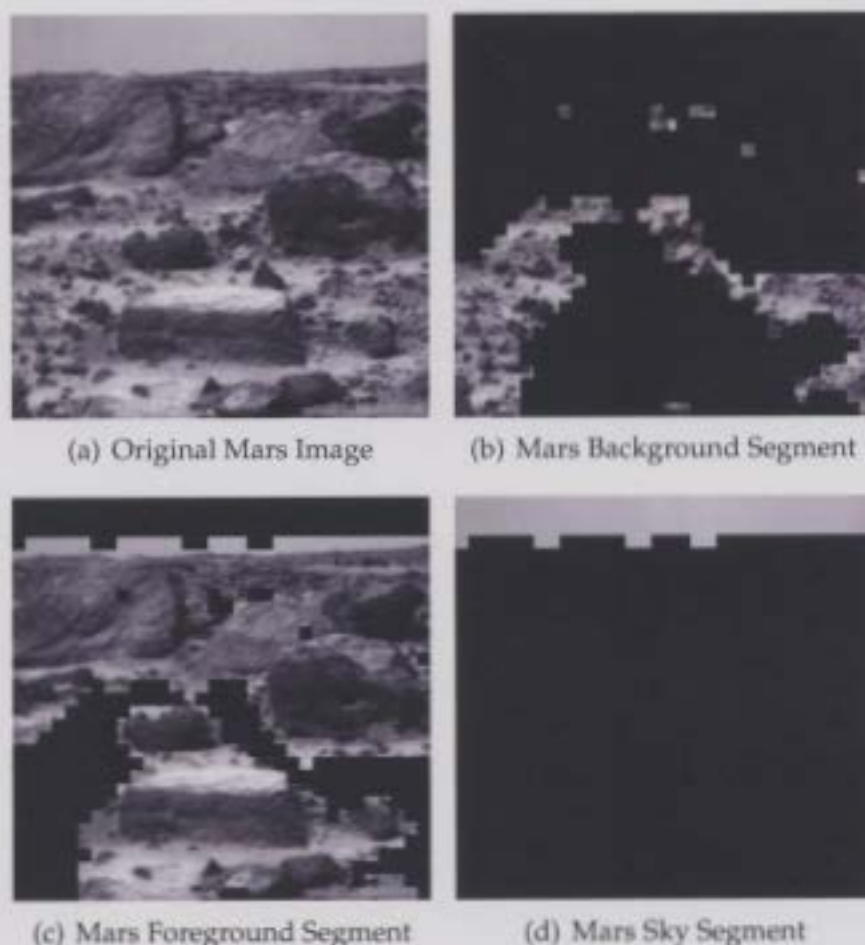


Figure 1.3: Segmentation of Mars Image into Sections that Can Be Compressed at Different Rates (from [16])

1.2.4 Unification of the Solution of Image Processing Problems

The just-described problems of image segmentation, denoising and compression can be unified and considered as a *molecule* of problems, in that they are each separate like *atoms* but can be combined together to form a larger unit. Though this may not be immediately obvious, the relatively new framework of image decomposition [1] unifies the solutions of these problems in providing a common framework to tackle all of them. Decomposition has in the past been used to refer to the splitting up of images into multiple resolutions in the context of wavelets [17]. Instead

the recent work by Meyer [1] is referred to in this thesis, where images are split into parts of differing priorities, which can be recombined by taking a sum to form the original image, all in the variational framework. The way that image decomposition can unify the three problems described above, and other image processing problems, is in providing a convenient representation of images for their joint solution. Next, some background on the problem of image decomposition is given.

1.3 Introduction to Image Decomposition

In his monograph, Meyer [1] talks about separating images into a higher priority and a lower priority component. He states that there is an inherent difficulty in deciding what information is considered to be of higher priority and what should be considered lower priority, but in general, the higher priority component is structured, while the lower priority component is more random, and can be described statistically. It can be easily seen that in general, there can be more than two priorities, e.g. high, medium and low.

Given that the original still-image information is split into these two components by their priority, if there is limited communication bandwidth, it is clear that the high priority component should generally be sent first, and then the lower priority component second. It is also possible to split an image into more than two components by their priorities. This is done for example when the acquisition of image data is corrupted by noise, which can be placed in a third component separate from structure and texture information.

In this thesis, attention is restricted to image decomposition models where the

high-priority component is piecewise smooth (sometimes called the cartoon component), and the lower-priority component(s) consist of texture/noise. A quick example of such a decomposition is shown in Figure 1.4 and described in the next section.

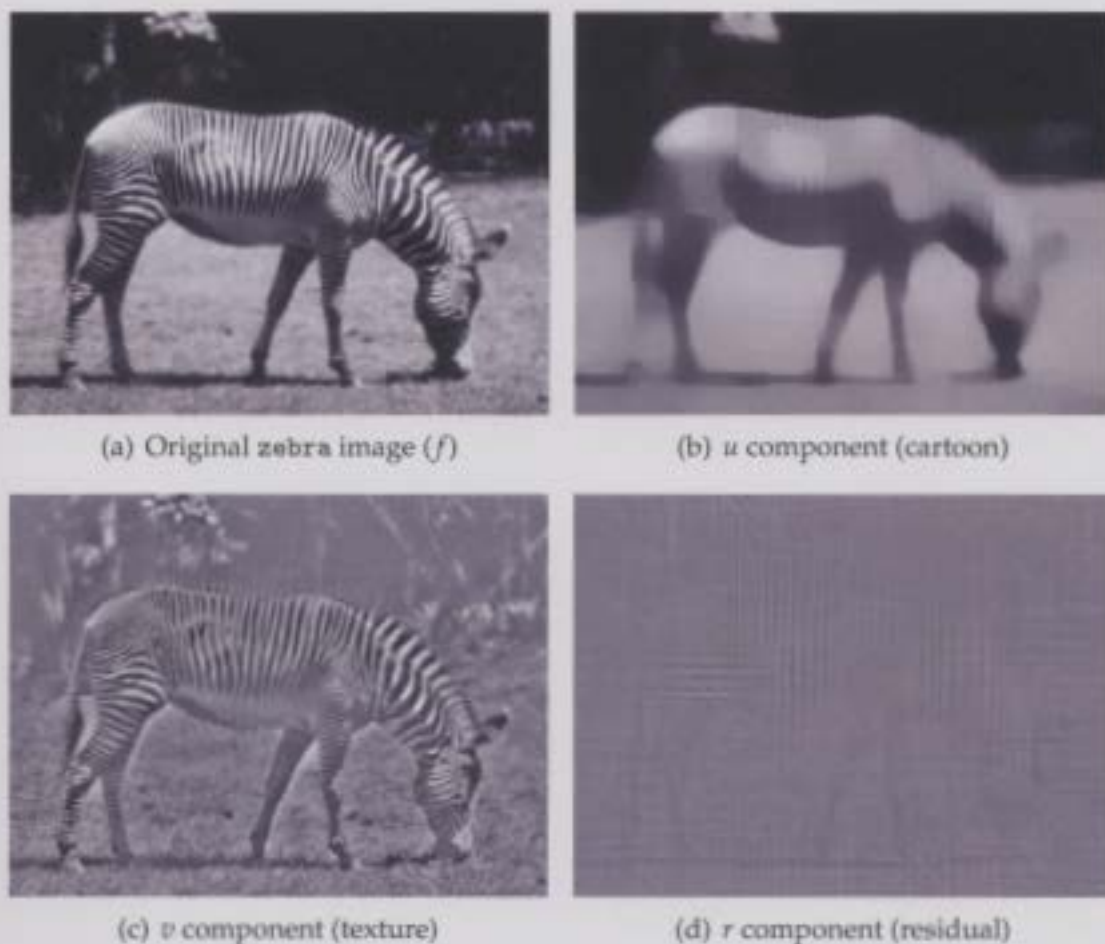


Figure 1.4: Sample Image Decomposition of zebra image

1.3.1 Example of Image Decomposition

An example decomposition, computed with the variational model of Vese and Osher [2], is shown in Figure 1.4. The Vese-Osher model is described in more de-

tail in Section 2.4.2. This zebra image in Figure 1.4(a) has been chosen in order to illustrate the image decomposition process because it contains objects of many different scales - for example the zebra is of large scale whereas the grass on the ground is very small-scale. So the zebra as well as the ground could be considered as objects in the cartoon component u , and the texture component v would fill these objects with details, e.g. the zebra's stripes and the blades of grass. As can be seen, the cartoon component consists of the shape of the zebra, which looks more like a horse and the grass and background have been smoothed, though the interface between them is still not completely blurred. The residual component r (which equals $f - u - v$) is very close to zero, the maximum deviation from which is about 8 percent of the dynamic range of the original zebra image. Note that 130 has been added to the texture and residual components for display purposes, where 255 is the maximum gray level in the images.

Next we discuss how the operation of image decomposition can be considered simply to be the transformation to an alternate representation of the information in an image.

1.4 Representation of Digital Images with Image Decomposition

In engineering and mathematics, it is often advantageous to change the representation of data to facilitate operations on the data. For instance, as mentioned in the book by Marr [18], the representation of numbers will be different depending

on the operations we wish to do on them. For common use, the regular decimal system is sufficient, allowing for efficient performance of such common operations as subtraction and multiplication by human beings with pen and paper. However, a binary representation is more well-suited to the two-level logic of a digital computer, so in this context, a number is represented by bits. If we were to use a binary representation for human use, then this would lead to very inefficient representations of numbers - for example, the 3-digit number 563 in decimal, would have to be represented by the 10-digit number, 1000110011 in binary.

Similarly, in the context of signal and image processing, some representations are more conducive to certain operations than others. The classical example is representing an image in the frequency domain. Conversion to the frequency domain can be performed very efficiently using the 2-D Fast Fourier Transform (FFT), after which frequency filtering can be efficiently performed in the frequency domain by pointwise multiplication. Subsequently, the filtered frequency-domain image can be converted back to the spatial domain by the inverse Fast Fourier transform which has computational complexity equal to that of the forward FFT, namely $O(N \log N)$ where N is the number of pixels in the image.

Another example of an image representation conducive to certain operations is that of colour image representation in various colour spaces. The most common colour space is the usual RGB colour space, where each pixel is represented by the three intensities of red, green and blue. The RGB representation is used for example in colour CRT and LCD monitors. On the other hand, for compression of RGB images, it is preferred to first transform to the YCbCr colour space, since the three colour planes in that space are much less correlated with each other than those for

the image in RGB space. Because of this reduced redundancy in the YCbCr representation, higher compression rates may be obtained [19]. Image decomposition is simply another representation of a digital image, in this case as the sum of two or more component images.

An interesting instance of image decomposition occurring in the natural world, is in the human psychovisual system. It has been demonstrated that patients with damage to one hemisphere of their brain draw different aspects of a picture shown to them for a short period of time, once the picture has been removed. They either draw the outline of the picture (if their left hemisphere is damaged) or the details (if their right hemisphere is damaged) [20]. The outline corresponds to the cartoon component, while the details correspond to the texture/noise component. To the author's knowledge, this relation between image decomposition and psychovisual function has not been observed before, and could be important to brain hemispheric modelling.

The representation of digital images as sums of components obtained from image decomposition is useful for many diverse image processing applications. In Meyer's monograph [1], decomposition was deemed to be important for image coding and transmission, with at least one practical implementation [21] appearing in the literature. In subsequent literature, decomposition has also proven to be useful for texture discrimination [2], image denoising [4], image inpainting [22], and image registration [23]. For example, simultaneous structure and texture image inpainting refers to reconstructing damaged or missing sections of an image from the image information surrounding the sections [22]. For this type of inpainting, an image is split into cartoon and texture components, and then inpainting

methods tuned to each component type are used.

Image decomposition can also be used for image denoising [4], because the v component consists of texture and/or noise. However, for a two-component image decomposition model, this is done at the expense of losing some texture information from the u component as well. Image denoising is one of the examples of the applications of image decomposition examined in Chapters 5 and 6 of this thesis.

1.5 How Decomposition is Applied - Sequential and Simultaneous Solutions

1.5.1 Simultaneous Solution of Image Processing Problems

Often, two or more image processing problems can be solved in tandem. The impetus for solving more than one problem simultaneously is not only that two problems are solved at once, but that the information from one problem can help in the solution of the other, and vice versa.

For example, if it is desired to simultaneously deblur (without knowing the blurring kernel) and segment an image, a segmentation of the image can help with this blind deblurring. This is because if the locations of the edges are known, then the shape of the blurring kernel can be determined by how the edge in the original image looks in the blurred image. Deblurring the image helps the segmentation process because then the edges of the objects in the image can be used in gradient-

based image segmentation (for non-textured images). An example of such a simultaneous deblurring/segmentation algorithm is found in [24].

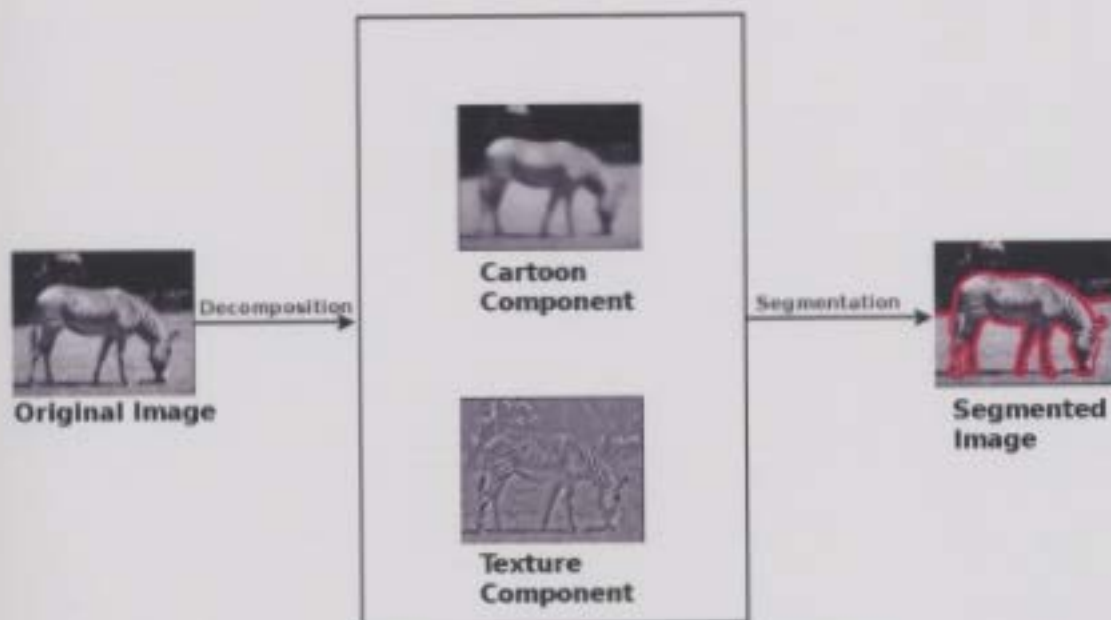
1.5.2 Sequential and Simultaneous Solutions in this Thesis

During this research, image decomposition was used in two different ways. The more common and standard way was to first apply image decomposition to obtain the component-by-component representation, and then after this, applying another image processing algorithm to this representation. A second way, was to simultaneously solve both the image decomposition problem and the other image processing problem by alternating between the solution of the two problems.

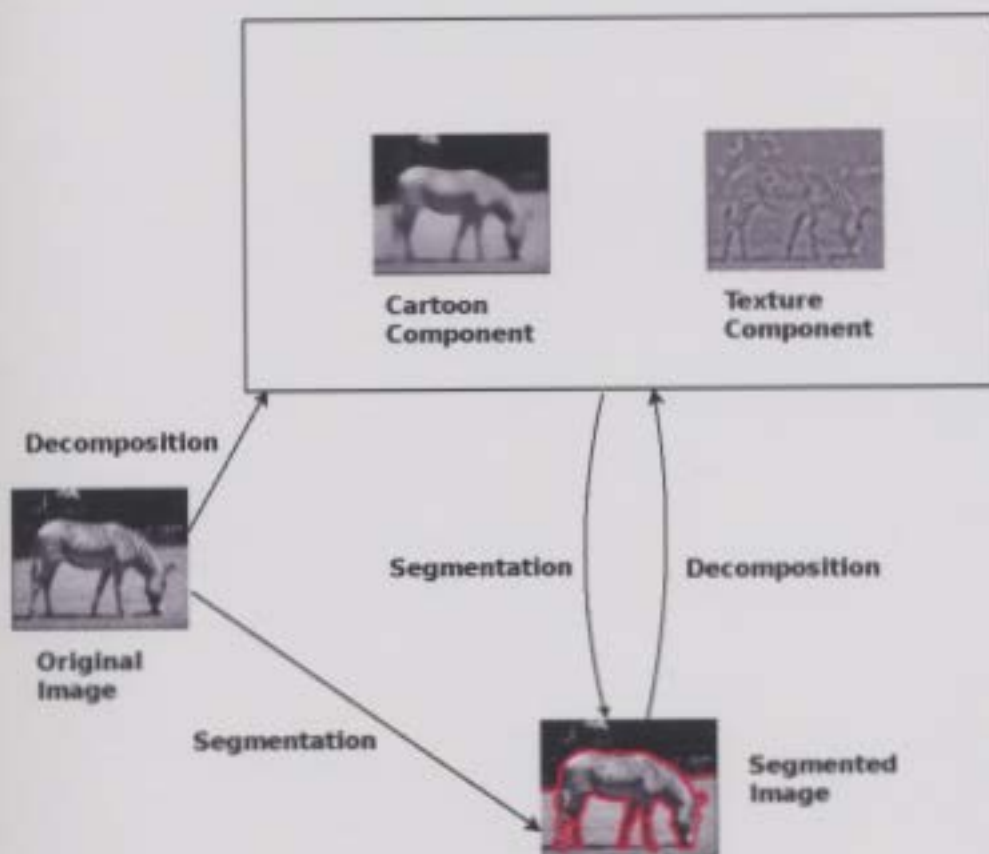
In Figure 1.5, examples of the sequential and simultaneous solutions of the image decomposition and image segmentation problems are shown. In this thesis, a simultaneous decomposition and segmentation scheme is developed in Chapter 4, as first published by Shahidi and Moloney in [25], and this fits into the simultaneous solution framework in Figure 1.5(b). However the scheme is more complex than the figure illustrates, in that there are texture subcomponents, and it is these subcomponents that the segmentation algorithm operates on, not the texture component directly.

1.6 Problem Statement

Current methods for image decomposition as described above generally use numerical partial differential equation techniques, more specifically the finite differ-



(a) Sequential Solution of Image Processing Problems



(b) Simultaneous Solution of Image Processing Problems

Figure 1.5: Flow Diagram of Sequential and Simultaneous Solutions of Image Decomposition and Another Image Processing Problem (Segmentation)

ence method, to minimize energy functionals.

One drawback of such variational image decomposition models is that, even on modern computers, they take a relatively long time, on the order of several minutes for some images. Although in this thesis, real-time/hardware implementations of the algorithms designed and used are not considered, this is a natural step to follow and an aim for future research.

Additionally, many decomposition models do not perform well on certain types of images, or for certain image structures. For example, in the paper of Vese and Osher [2], the results from decomposition are input to an image discrimination method, but because sometimes the quality of decomposition is reduced, mainly by the presence of high-priority edges in the lower-priority component, so is the quality of the ensuing discrimination. As well, in a previous paper, Osher, Solé and Vese [4] apply their decomposition model to the problem of textured image denoising. However, for small-scale texture, often image structure is placed in the low-priority noise component, which is not desirable.

Thus the problem that is being tackled in this thesis is the high-quality and efficient decomposition of images to simultaneously solve other image processing and analysis problems, e.g. texture denoising and discrimination/segmentation or inpainting.

The emphasis in this thesis is on the image decomposition problem itself, but some applications to the other image processing problems to which it can be applied are also included.

1.7 Proposed Method of Solution

As stated in the previous section, existing variational image decomposition models often suffer from poor decomposition quality or efficiency of solution. Hence, these models are used as a base, and extended so these drawbacks are addressed. In general, the energy functionals for the various decomposition models are modified, either by adding extra terms to the functionals, their associated partial differential equations, or by changing the terms that are already present.

When extra terms are added to these functionals, additional desirable conditions are imposed within and between the components of the decompositions. An example is to not to allow texture and cartoon edges to be strong at the same image pixels. Another example is to add a term to the PDEs to be solved to promote diffusion along image isophotes in order to more effectively use the decomposition model for oriented texture denoising.

1.8 A Roadmap and Original Contributions of this Thesis

There are many existing image decomposition models, though it has been found that most are not as efficient as desirable. The aim of image decomposition is to separate the structure of an image from the texture, but it has been discovered in the course of this research that some of the implementations of existing models don't do this as cleanly as one would want. So these models have been modified by adding extra terms to the defining energy functionals to achieve both quality

and efficiency goals. New models have also been created for new applications, for example the denoising of oriented texture. In Chapter 3, the first modified model is presented, called Improved Edge Segregation (IES) , the main aim of which is to improve the quality of the existing Vese-Osher model, by placing more of the cartoon edges in the original image into the computed cartoon component and more of the small-scale edges into the texture component. Also in Chapter 3, the first, to our knowledge, implementation of the solution of the image decomposition problem in the Message Passing Interface on a parallel computer is described.

It is possible to view the decomposition and segmentation of images as two problems which can be naturally solved together, and this is done in Chapter 4 of this thesis, with the new Simultaneous Decomposition/Discrimination (SDD) model. This greatly improves the efficiency of these two tasks in terms of number of iterations required over the usual sequential implementation, and gives better quality results as well.

The strictly two-component decomposition model, called in this thesis the Osher-Solé-Vese model after the original authors [4], is also modified to create three new models. These modifications fall under two categories, and these are described in Chapter 5. One modified model, which is referred to as the Decorrelated Osher-Solé-Vese (DOSV) model, decorrelates the cartoon and texture component by explicitly adding a term based on the local correlation coefficient between these two components. Another modified model, referred to in this thesis as the Perona-Malik-Osher-Solé-Vese (PMOSV) model, combines the nonlinear diffusion framework of Perona and Malik with the decomposition model of Osher, Solé and Vese. This model has some drawbacks, which leads to the introduction of the Oriented

Laplacian Osher-Solé-Vese (LOSV) model, also based on the idea of combining decomposition with nonlinear diffusion. Although the Osher-Solé-Vese model has been claimed by the authors to be very suitable for denoising of texture, it is shown here that the LOSV model performs even better for certain images with very high frequency texture present.

In Chapter 6, the new decomposition model called Orientation-Adaptive Decomposition (OAD) is introduced. This model is especially suitable for the decomposition of oriented texture; hence, one of its applications is for the denoising of such textures. This model only uses one subcomponent which needs to be calculated at each iteration to represent the texture, and so is simpler to understand and implement. An extension called Eikonal Orientation-Adaptive Decomposition (EOAD) is explored as well in that chapter. Finally, Chapter 7 summarizes the thesis, makes some conclusions, and suggests possible future work stemming from this thesis.

CHAPTER 2

A Review of Variational Image Processing Models and Other Necessary Tools

2.1 Introduction

In the introductory chapter, variational methods for image processing and image decomposition were briefly touched upon, e.g. in Section 1.6. In this chapter, the Energy Method is reviewed, which is used for solving image processing problems within the variational framework, and some of the existing variational image decomposition models in the literature presented.

The decomposition and other variational models in this thesis use many image processing tools which are mathematical in nature, many of which are used repeatedly. So a description of these image processing tools needed for the various new

decomposition models and associated problems, e.g. texture discrimination/segmentation, are also presented in this chapter.

2.2 Background: Nonlinear PDE Models in Image Processing

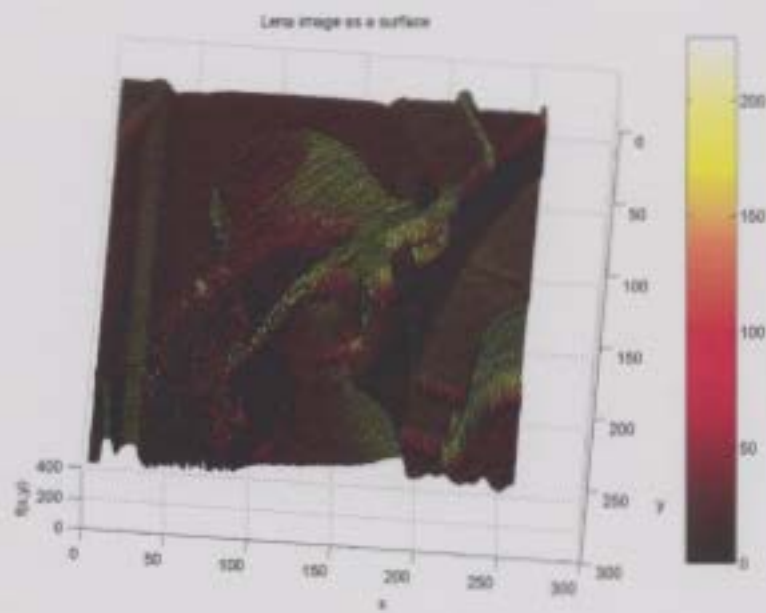
In the literature, PDE methods have been used for practically every image processing and computer vision application, including continuous mathematical morphology, invariant shape analysis, shape from shading, segmentation, object detection, optical flow, stereo, image denoising, image sharpening, contrast enhancement and image quantization [26].

Such methods were popularized with the introduction of scale space by Koenderinck [27] and Witkin [28] in 1983/1984. The possibility of the use of a multiscale framework for image processing is motivated by the fact that in the real world, objects can be observed and photographed at a variety of scales. A common example is that of a tree. From afar, only the main shape and outline are visible. Then a little closer, the shapes of the leaves and the ridges on the trunk can be seen. Extremely close, at a microscopic level, the individual cells of the tree can be distinguished. All of these scales describe the object, as they can be captured in a digital image; however only a certain range of scales can be observed in any given picture.

In PDE-based methods for image processing, the digital image is considered to be a continuous three-dimensional surface, with the height at each point of the surface equal to the intensity of the image at the corresponding pixel or picture



(a)



(b)

Figure 2.1: (a) Original Lena image and (b) Lena image as a surface

element (at least for grayscale or other one-channel images). An example is shown in Figure 2.1. The conversion to a continuous 3-D surface is implicit in any PDE-based image processing method because, theoretically, PDEs must operate on a continuous surface, and not on a regularly sampled surface, which is what defines a digital image.

2.2.1 Example of the Energy Method in PDE-based Image Processing

2.2.1.1 The Energy Method

An example is now given of how the Energy Method for PDE-based image processing proceeds for a simple problem. Digital image processing problems consist of taking an input digital image (with domain and range elements quantized to discrete quantities), and then processing it to create one or more output images. The processed image may be enhanced in some way (for example by being contrast enhanced or deblurred), and/or the information in the image changed in some way to a more useful representation (for example by applying the wavelet transform).

The first step in the Energy Method is to consider an image (assumed to consist of one colour channel for the purposes of this thesis) to be a first-order continuously differentiable function $I : \Omega \subseteq \mathbb{R}^2 \rightarrow \mathbb{R}$, and then to define an energy on the class of such images. By a first-order continuously differentiable function, it is meant a function which is differentiable, and for which the first-order partial derivatives are continuous. The domain of the image is Ω , which is usually rectangular since standard digital images are of this shape. This is opposed to the

original representation of the digital image as an $M \times N$ matrix of either bounded discrete quantized levels (e.g. bytes), or bounded values in \mathbb{R} (e.g. floating point values). The energy is defined to be a functional, which takes as its inputs one or more first-order continuously differentiable functions and their partial derivatives and gives a real number as its output. The energy is designed so that it is lower when its argument image functions have desired properties and is higher when they do not. So a desired processed image will be obtained when this energy functional is minimized. Because such energies are usually defined in terms of the integral of the argument functions and their partial derivatives, this minimization can be accomplished by the calculus of variations. Variational calculus shows that the minimizer of an energy functional satisfies the Euler-Lagrange equation(s), which is a set of one or more PDEs that the functions(s) minimizing the energy must satisfy. Given an energy functional, e.g.

$$\mathbf{E}(I, I_x, I_y) = \int_{\Omega} F(I, I_x, I_y) dx dy, \quad (2.2.1)$$

where I_x and I_y are the partial derivatives of I , we say that $\mathbf{E}(I, I_x, I_y)$ is convex if

$$\mathbf{E}(\lambda \vec{I}_1 + (1 - \lambda) \vec{I}_2) \leq \lambda \mathbf{E}(\vec{I}_1) + (1 - \lambda) \mathbf{E}(\vec{I}_2), \quad (2.2.2)$$

for all $\lambda \in [0, 1]$, where $\vec{I}_1 = (I_1, I_{1,x}, I_{1,y})$ and $\vec{I}_2 = (I_2, I_{2,x}, I_{2,y})$ are two vector-valued functions, with the I_1 and I_2 element functions of \vec{I}_1 and \vec{I}_2 being themselves two first-order continuously differentiable image functions [29].

Then if the energy functional $\mathbf{E}(I, I_x, I_y)$ in Equation 2.2.1 is convex, and F satisfies certain inequalities (e.g. bounds on the growth of partial derivatives of F) [29],

it can be shown that $\mathbf{E}(I, I_x, I_y)$ has a minimum, and the image I_{\min} minimizing $\mathbf{E}(I, I_x, I_y)$ is given by the following equation

$$\mathbf{E}'(I, I_x, I_y) = \frac{\partial F}{\partial I} - \frac{d}{dx} \left(\frac{\partial F}{\partial I_x} \right) - \frac{d}{dy} \left(\frac{\partial F}{\partial I_y} \right) = 0, \quad (2.2.3)$$

when substituted for I , I_x and I_y . Equation 2.2.3 is called the Euler-Lagrange Equation of the energy $\mathbf{E}(I, I_x, I_y)$. If the energy functional $\mathbf{E}(I, I_x, I_y)$ depends on higher order partial derivatives of I , then the Euler-Lagrange equation will change; however for the purposes of this thesis, only functionals which depend solely on the input image and its first-order partial derivatives are considered.

It is also possible (See Chapter 6 of this thesis) that there may be constraints placed on the image I , to force the final processed image to have certain desired properties. If it is wished to minimize the energy in Equation 2.2.1 subject to the constraint that $G(I, I_x, I_y) = 0$ everywhere, then the minimum of the constrained functional can be found by the method of Lagrange multipliers. If the space-varying Lagrange multiplier $\lambda(x, y)$ is introduced, then the minimum of Equation 2.2.1 subject to the constraint $G(I, I_x, I_y) = 0$, is found by solving the Euler-Lagrange equations of the energy

$$\mathbf{E}(I, I_x, I_y) = \int_{\Omega} [F(I, I_x, I_y) + \lambda(x, y)G(I, I_x, I_y)] dx dy. \quad (2.2.4)$$

In order to be able to minimize the above constrained energy in Equation 2.2.4, the same conditions that had to hold for $\mathbf{E}(I, I_x, I_y)$ and F in order for the solution of the Euler-Lagrange equation in Equation 2.2.3 to give a valid minimizer must hold, except that now the conditions must hold for the new $\mathbf{E}(I, I_x, I_y)$, and $F + \lambda G$

instead of F .

There will be an extra equation for the new function $\lambda(x, y)$, which has to be solved. The complete set of Euler-Lagrange equations is

$$\begin{cases} \frac{\partial F}{\partial I} + \lambda \frac{\partial G}{\partial I} = \frac{d}{dx} \left(\frac{\partial F}{\partial I_x} + \lambda \frac{\partial G}{\partial I_x} \right) + \frac{d}{dy} \left(\frac{\partial F}{\partial I_y} + \lambda \frac{\partial G}{\partial I_y} \right) \\ G(I, I_x, I_y) = 0 \end{cases}$$

Next, an artificial time parameter t is introduced to represent the process of evolution towards the energy minimum, so that $I(x, y)$ becomes $I(x, y, t)$, and the input image is set to be $I(x, y, 0) = I_0(x, y)$.

Often, gradient descent with explicit timestepping is used to solve the Euler-Lagrange equation. This means that a small time step Δt is introduced, and the following equation iterated

$$I^{n+1} = I^n - \Delta t \mathbf{E}'(I, I_x, I_y).$$

If the time step is small enough, then it can be shown that the above equation converges to a local minimum or a saddle point of the functional. With an appropriate choice of initial condition for I , the gradient descent equation will converge to a global minimum of the functional $\mathbf{E}(I, I_x, I_y)$, should one exist. The main drawback of explicit timestepping is that according to a condition called the Courant-Frederichs-Levy (CFL) condition for hyperbolic equations [29], often the time step has to be extremely small leading to a slow solution of the Euler-Lagrange PDE. Similar upper bounds on the time step exist for the other classes of PDEs, namely elliptic and parabolic [30]. The entire procedure is now illustrated with the follow-

ing example.

2.2.1.2 Example of the Energy Method - the Heat Equation

Continuous Heat Equation As a simple demonstration of the Energy Method, consider the following energy functional to be minimized

$$\mathbf{E}_{\text{heat}}(I, I_x, I_y) = \frac{1}{2} \int_{\Omega} |\nabla I|^2 dx dy = \frac{1}{2} \int_{\Omega} (I_x^2 + I_y^2) dx dy \quad (2.2.5)$$

Clearly, this functional favours images I which are smooth, and the global minimizer is a constant image. Using Equation 2.2.3 with $F(I, I_x, I_y) = (I_x^2 + I_y^2)$, it is seen that the minimizer of $\mathbf{E}_{\text{heat}}(I, I_x, I_y)$ satisfies the equation $I_{xx} + I_{yy} = 0$. Because of the simplicity of this equation, it can be solved analytically, but instead this example is continued with the general methodology for the Energy Method.

An artificial time parameter is introduced, so that now the equation

$$I_t = I_{xx} + I_{yy} = \Delta I, \quad (2.2.6)$$

is obtained, where $I(x, y, 0) = I_0(x, y)$, $I_0(x, y)$ being the initial input image. Clearly, when this PDE reaches steady-state ($I_t = 0$), the original energy functional will be minimized.

It has been shown by Koenderink [27] that evolution of an image by this heat equation is equivalent to convolution with a Gaussian - in other words the input image $I_0(x, y)$ is isotropically blurred. As the time t increases, the variance of the Gaussian also increases, which means that there is more blurring and so the image

has a coarser resolution. In fact, as stated in [31], the variance of this Gaussian, as related to the time that diffusion is allowed to progress, is: $\sigma^2(t) = 2t$.

Discretization of Heat Equation The above example used the theoretical continuous image $I_0(x, y)$ and the continuous processed output images of the Energy Method. However, digital images are not continuous so it is necessary to consider the numerical form of the Energy Method. In order to solve the PDE numerically, the digital image is brought back to its original representation as an $M \times N$ quantized or real-valued matrix. The PDE(s), which is/are discretized in time and space, yield(s) the processed image after iterative evolution in (artificial) time. Finite differences are generally used for the discretization because of their simplicity and also due to the fact that any digital image is composed of regularly spaced or sampled pixels, for which finite differences are a natural implementation, as opposed to, for example, finite elements.

The art in this approach is choosing an appropriate energy functional for a given problem guided by the science of selecting features in the processed image which are desired to be maintained, enhanced or removed, and solving the PDE(s) using the appropriate numerical methods.

Continuing with the example of the heat equation, assuming that $I_{i,j}^k$ is the value of the digital image I at the i^{th} column and j^{th} row and at the k^{th} iteration, the iterative finite difference equation derived from Equation 2.2.6 that is obtained is

$$\begin{aligned} I_{i,j}^{n+1} &= I_{i,j}^n + \Delta t(I_{i-1,j}^n + I_{i,j-1}^n + I_{i,j+1}^n + I_{i+1,j}^n - 4I_{i,j}^n) \\ &= (1 - 4\Delta t)I_{i,j}^n + \Delta t(I_{i-1,j}^n + I_{i,j-1}^n + I_{i,j+1}^n + I_{i+1,j}^n) \end{aligned} \quad (2.2.7)$$

This equation uses discretized second order derivatives, e.g. $I_{xx} = \frac{I_{i+1,j} - 2I_{i,j} + I_{i-1,j}}{h^2}$ to approximate the partial derivatives in the continuous heat equation. Here, h is the grid spacing between adjacent entries in the matrix representing the image, and as done here, is generally taken to equal 1. Notice in the above equation, the value of any pixel (picture element, or entry in the digital image matrix) at the $(n + 1)^{st}$ iteration, only depends on the pixel values at the n^{th} iteration. This type of discretization is called Jacobi iteration. Its advantage is that all the values of the pixels at a given iteration are independent of each other and can be computed in parallel. One disadvantage is slower convergence as opposed to more advanced methods, e.g. the Gauss-Seidel iteration that is discussed below. Another disadvantage is that storage is necessary both for the image data of one iteration, and the next. It can be shown using eigenvalue analysis of the underlying matrix formulation of the discrete heat equation that this solution is only stable for time steps $\Delta t < \frac{1}{4}$ [30].

If the new pixel values in the image being processed are computed in row-major order, then some of the gridpoint (pixel) values from the current iteration will be available to be used for calculation of other gridpoint values of the same iteration. The expression for a method called Gauss-Seidel iteration, assuming that the progression being made is in row-major order, is given by

$$I_{i,j}^{n+1} = I_{i,j}^n + \Delta t (I_{i-1,j}^{n+1} + I_{i,j-1}^{n+1} + I_{i,j+1}^n + I_{i+1,j}^n - 4I_{i,j}^n) \quad (2.2.8)$$

This method can also be described by the pseudocode at the top of the next page.

As can be seen, one advantage of Gauss-Seidel over Jacobi iteration is that extra

Algorithm 1 Gauss-Seidel Iteration for Heat Equation

```
1: procedure GAUSS-SEIDEL ITERATION( $I$ )
2:   for  $n \leftarrow 1, numiters$  do
3:     for  $i \leftarrow 1, rows$  do
4:       for  $j \leftarrow 1, cols$  do
5:          $I_{i,j}^{n+1} = I_{i,j}^n + \Delta t(I_{i-1,j}^{n+1} + I_{i,j-1}^{n+1} + I_{i,j+1}^n + I_{i+1,j}^n - 4I_{i,j}^n)$ 
6:       end for
7:     end for
8:   end for
9: end procedure
```

storage is not required — image values are overwritten in place. Also, Gauss-Seidel iteration gives faster convergence. The disadvantage, however, is the fact that it is difficult to parallelize the iteration very well due to dependence between gridpoints of the same iteration. There is more on the comparison between Jacobi and Gauss-Seidel iteration in Section 3.7.2.

Illustration of Heat Equation An example of the heat equation applied to a tomographical image is shown in Figure 2.2.

As can be seen in Figure 2.2, as time increases, less detail can be seen in the evolving image I . Thus, the collection of images $\{I(x, y, t)\}_{0 \leq t \leq T}$ can be regarded as a linear scale space for use in a multiscale framework. Evolution of an image with the heat equation can be used for denoising, since noise is small-scale, and will dissipate with blurring. In digital images, however, edges, or regions of rapid change, carry the most information because they often indicate the boundaries of image objects. The utility of the linear scale space, described above, for denoising is hence limited by the lack of localization of edges. All parts of the image, edges and non-edges, are blurred by the same amount, so that some spatial information



(a) Original image I_0



(b) Image $I(x, y, t)$ evolved up to $t = 2$ with heat equation



(c) Image $I(x, y, t)$ evolved up to $t = 5$ with heat equation

Figure 2.2: Example of Isotropic Linear Diffusion

about the location of image objects is lost. This limitation of linear scale space and of linear diffusion leads to the notion of inhomogeneous diffusion, which is described next.

2.2.1.3 Perona-Malik Inhomogeneous Diffusion

It was only when Perona and Malik [5] developed variable conductance diffusion that the use of PDEs in image processing could be said to have become very widespread. They called their scheme anisotropic diffusion, though this is somewhat of a misnomer, as Weickert [32] explains and as will soon be discussed in the next paragraph. Perona and Malik introduced an edge-stopping coefficient to linear diffusion which ensures that only diffusion within regions which have low or zero edge-magnitude takes place and diffusion doesn't occur near strong edges. The implicit assumption of such diffusion is that the image is composed of non-textured smooth objects. Perona and Malik's diffusion is defined by a non-linear PDE and is a very powerful denoising tool.

The isotropic diffusion of the previous subsection could be expressed as $I_t = \text{div}(\nabla I)$, where div is the divergence operator and measures the diffusion of its argument. Perona and Malik [5] introduced an edge-stopping coefficient inside the divergence, so that their evolution equation is

$$I_t = \text{div}(g(|\nabla I|^2)\nabla I), \quad (2.2.9)$$

where $g(|\nabla I|^2)$ is the diffusivity with g being a non-increasing function. Therefore, when the image gradient magnitude $|\nabla I|$ is high, $g(|\nabla I|^2)$ should be low,

because the edge should not be diffused. Similarly, when the image gradient magnitude $|\nabla I|$ is low, more diffusion is desired, so $g(|\nabla I|^2)$ should be high. Thus, this diffusion is inhomogeneous because it varies with image location depending on the gradient strength of the evolving image. It is isotropic however, and not anisotropic, because at any given image location, diffusion is constant along all directions. Anisotropic diffusion can be accomplished by the use of a diffusion tensor [32] instead of a single diffusivity function.

There are many possible choices for $g(|\nabla I|^2)$; in [5], Perona and Malik primarily used

$$g(|\nabla I|^2) = \frac{1}{1 + \frac{|\nabla I|^2}{\lambda^2}},$$

where λ is the diffusivity parameter. Although Perona and Malik did not develop their method via minimization of an energy functional, and instead looked at the structure of the image being processed, Perona-Malik (P-M) diffusion fits within the energy minimization framework, described at the start of Section 2.2.1. As stated in [32], if the energy E_{PM} is defined as

$$E_{\text{PM}}(I) = \int_{\Omega} \frac{\lambda^2}{2} \ln \left(1 + \left(\frac{|\nabla I|}{\lambda} \right)^2 \right) dx dy,$$

then the derived Euler-Lagrange equation will be the same as Equation 2.2.9.

An example of P-M diffusion on the same image as Figure 2.2(a) is shown in Figure 2.3, with diffusivity parameter $\lambda = 8$ (assuming that the maximum gray level in the image is 255).

Notice that the diffused images in Figures 2.3(b) and 2.3(c) are less blurry than



(a) Initial Image I_0



(b) Image $I(x, y, t)$ evolved up to $t = 2$ with P-M diffusion



(c) Image $I(x, y, t)$ evolved up to $t = 5$ with P-M diffusion

Figure 2.3: Example of Inhomogeneous Perona-Malik Diffusion

the corresponding diffused images in Figure 2.2. It can be seen that the P-M diffusion keeps the large scale edges in the initial image so that the information provided by these edges is not lost, though there is some smoothing within regions of the tomographical image. This makes Perona-Malik diffusion better suited than isotropic homogeneous diffusion for scale space formation, and for use with images which are typically strongly characterized strongly by their edges.

2.2.1.4 Summary

The general approach being thus undertaken in this research is a common one for PDE-based image processing – first, an energy functional is defined on the image, generally an integral involving the image intensities and gradients, and the minimum of this energy found (i.e. the solution of the Euler-Lagrange equation(s)) using the calculus of variations [33]. The expression in the Euler-Lagrange equation is set to the derivative of the image with respect to an artificial time parameter, yielding a PDE or a system of PDEs, and the image is allowed to evolve (in theory continuously, in practice numerically) until it reaches a steady state. The continuous steady state is a solution to the Euler-Lagrange equation(s), and thus the desired energy minimum (assuming certain conditions hold on the energy and its arguments). The PDE(s), which is/are discretized in time and space, yield(s) the processed image (the numerical steady state) after evolution in time.

The advantage of using PDE models vs. traditional methods in image processing such as linear or non-linear filtering is that there is a very broad and rigorous theory of PDEs [34] that has already been developed for other purposes, and uniqueness and convergence of various schemes can be proven using this theory.

Also, such methods are sometimes of higher speed and are often more flexible and powerful than other forms of filtering.

2.3 Additive Operator Splitting

As Additive Operator Splitting (AOS) is used several times in this thesis, it is explained in some detail here for the purposes of completeness. In one dimension, for a signal $u(x)$, general nonlinear diffusion takes the form

$$u_t = \partial_x [g(|\partial_x u_\sigma|^2) u_x]. \quad (2.3.1)$$

In the above equation, g is the nonlinear diffusivity function, and u_σ is the image u blurred with a Gaussian of standard deviation σ . This blurring is performed to make the model robust to noise and to make the problem well-posed [35]. When compared with the expression in Equation 2.2.9, it may be observed that the above equation is the one-dimensional analogue of the 2-D form of that equation. If u_i^k is set to be the value of u at position x_i and at time $t_k = k\Delta t$, then this equation can be discretized explicitly as follows

$$u_i^{k+1} = u_i^k + \Delta t \sum_{j \in \mathcal{N}(i)} \frac{g_j^k + g_i^k}{2} (u_j^k - u_i^k), \quad (2.3.2)$$

with $\mathcal{N}(i)$, the set of pixels neighbouring pixel i , and g_i^k approximating the nonlinear diffusivity function at position x_i and time t_k . The formula for g_i^k is

$$g_i^k = g \left[\frac{1}{2} \sum_{p,q \in \mathcal{N}(i)} \left(\frac{u_p^k - u_q^k}{2} \right)^2 \right]. \quad (2.3.3)$$

This reduces Equation 2.3.2 to the matrix equation $u^{k+1} = [I + \Delta t A(u^k)]u^k$, where the elements of matrix $A(u^k) = [a_{ij}(u^k)]$ are given by

$$a_{ij}(u^k) = \begin{cases} \frac{g_i^k + g_j^k}{2} & \text{for } j \in \mathcal{N}(i) \\ -\sum_{n \in \mathcal{N}(i)} \frac{g_i^k + g_n^k}{2} & \text{if } j = i \\ 0 & \text{otherwise} \end{cases}$$

It is proven in [36] that there is a rather severe time step restriction of $\Delta t < \frac{1}{2}$, so instead of explicit time stepping, it is proposed that semi-implicit time stepping be used instead. So, in lieu of the equation $u^{k+1} = u^k + \Delta t A(u^k)u^k$, the equation $u^{k+1} = u^k + \Delta t A(u^k)u^{k+1}$ is solved. Because it can be seen from its definition that the matrix $A(u^k)$ is tridiagonal, and since it can be shown easily that $A(u^k)$ is diagonally dominant, the semi-implicit equation $u^{k+1} = (I - \Delta t A(u^k))^{-1}u^k$ where I is the identity matrix, can be solved using the Thomas algorithm [36].

This is a very simple algorithm which is linear in the number of points in the solution domain. It consists of first performing an LR decomposition [37] of the tridiagonal matrix $B = (I - \Delta t A(u^k))^{-1}$, thus factoring it into the product of an upper bidiagonal and lower bidiagonal matrix ($B = LR$). Then $LRu = d$ can be solved by forward and backward substitution, first by solving $Ly = d$, and then

$$Ru = y.$$

The above procedure is valid for one-dimensional problems. For higher dimensional problems, especially for the 2-D problems which are considered in this thesis, the above procedure doesn't generalize directly, and some modifications have to be made to the semi-implicit scheme described above. For two dimensions, the nonlinear diffusion equation becomes

$$u_t = \partial_{x_1}(g(|\nabla u_\sigma|)\partial_{x_1}u) + \partial_{x_2}(g(|\nabla u_\sigma|)\partial_{x_2}u) \quad (2.3.4)$$

The corresponding semi-implicit equation becomes

$$u^{k+1} = (I - \Delta t \sum_{l=1}^2 A_l(u^k))^{-1} u^k \quad (2.3.5)$$

The matrix $A_1(u^k)$ corresponds to the derivative matrix in the x direction, and $A_2(u^k)$ to the derivative matrix in the y direction. Here the image at any iteration k is rewritten as a vector u^k traversing the image matrix in row-major order.

Unfortunately, in two dimensions, there is no ordering of the pixels that will ensure that the bandwidth of the matrix

$$I - \Delta t \sum_{l=1}^2 A_l(u^k)$$

will be small. Thus even for the semi-implicit solution scheme, it will be impossible to use the Thomas algorithm to solve each system. However, the semi-implicit

equation can be modified to form a new system [36]

$$u^{k+1} = \frac{1}{2} \sum_{l=1}^2 [I - 2\Delta t A_l(u^k)]^{-1} u^k \quad (2.3.6)$$

It can be easily shown that both the original and modified semi-implicit equations have the same first-order Taylor approximations. The advantage of the new system is that once again the Thomas algorithm can be applied because each of the A_l 's is tridiagonal. This is called the Additive Operator Splitting scheme, and is highly efficient. It has been proven that the scheme is unconditionally stable, meaning that it will not grow unboundedly, regardless of the value of the time step Δt . Only the accuracy of the solution is affected as Δt is increased.

2.4 Image Decomposition Models - Mathematical Underpinnings

2.4.1 Background Image Decomposition Theory

To approach the simultaneous solution of two or more image processing problems, textured image decomposition has been examined. Decomposition is a common first step towards texture denoising and discrimination. Having the output of an image decomposition can lead to effective denoising of a digital image, and, as well, can make efficient texture segmentation/discrimination much more practical. In a broad sense, image decomposition is defined by the separation of an image into two or more components which can be combined in some way to obtain the

original image.

Often, the image f is split into the sum of two components (e.g. $f(x, y) = u(x, y) + v(x, y)$), where $u(x, y)$ models the objects in the image and $v(x, y)$ contains the texture and/or noise in the image $f(x, y)$.

In most previous approaches, u and v are modelled as belonging to specific Banach spaces (complete normed vector spaces). For example the component u is assumed to belong to the space BV of functions of bounded variation (meaning the total integral of the gradient magnitude over the domain of the function is bounded above). This is because BV has been found in e.g. [38], to model piecewise smooth functions well. The image $u(x, y)$ is often restricted to be piecewise smooth, and is thus often referred to as the cartoon component, since the u image looks like a cartoon you would see in the comics section of a newspaper, or an animated cartoon on Saturday morning television. The component v is what is left over and is the texture/noise component. There are also three-component models (see e.g. Aujol et. al. [39]), where the texture and noise are divided into two separate components, v and w respectively.

In practice, because the decomposition is usually not exact, there is a residual component r which tends to be very small. Therefore, the two-component model can be expressed with the equation $f(x, y) = u(x, y) + v(x, y) + r(x, y)$.

Recently, Daubechies and Teschke [40] have proposed an image decomposition method based on Meyer's decomposition of images into cartoon, texture and noise components [1]; their method combines both wavelet and variational frameworks. This algorithm actually performs simultaneous deblurring and denoising of tex-

tured images, and does this more efficiently than Malgouyres' variational algorithm [41]. However, Daubechies' and Teschke's algorithm is mathematically very complex. Also, their method assumes that the blurring kernel is known. This could be extended to include blind deblurring, for example by adding a total variation term for this kernel as in [42], so that is a possibility which could be studied in the future. Meyer also considers the wavelet shrinkage procedure of Donoho when applied to functions of bounded variation, and functions belonging to other Besov spaces [1].

As just mentioned, various researchers, e.g. Meyer [1], have proposed or used a decomposition of an image f into a sum $u + v$ where u is a cartoon component of bounded variation and v is an oscillating component consisting of texture and/or noise.

More generally, the usual total variation (TV) flow results from minimizing the energy functional

$$\mathbf{E}_{\text{TV}}(u) = \int_{\Omega} |\nabla u| dx dy + \lambda \int_{\Omega} (f - u)^2 dx dy. \quad (2.4.1)$$

The first term ($\int_{\Omega} |\nabla u| dx dy$) is meant to produce a bounded variation (piecewise-smooth) image upon energy minimization, while the second term is a fidelity term, which ensures that the result is close to the initial image f . This functional was first introduced in an image processing context in [38]. Although the minimization of $\mathbf{E}_{\text{TV}}(u)$ preserves sharp edges, it destroys fine structure such as texture. However, the flow has been used successfully for the denoising and deblurring of images of bounded variation.

In addition, in [1], Meyer proposed changing the second term in the above energy from including the L^2 -norm of the residual to the $*$ -norm of this residual, where $*$ is a norm defined on a suitably defined Banach space G . The norm on G is defined by

$$\|v\|_* = \inf_{g_1, g_2} \left\| \sqrt{g_1^2(x, y) + g_2^2(x, y)} \right\|_{L^\infty}, \quad (2.4.2)$$

over all g_1 and g_2 such that $v = \text{div}(\vec{g})$ where $\vec{g} = (g_1, g_2)$. In other words, the above L^∞ -norm (supremum) must be minimized over all g_1 and g_2 such that

$$v = g_{1,x} + g_{2,y}. \quad (2.4.3)$$

It is difficult in the literature to find an intuitive explanation of how or why this norm models texture and/or noise in an image, however it has been shown that the norm is small for functions which may have large oscillations. In [1], it is proven in a lemma that if the following properties hold

1. There exists a sequence of functions $\{f_n\}_{n=1}^\infty$, which are in $L^2(D)$, with D a disc, and the supports of f_n contained in a compact set $K \subseteq D$.
2. The L^q norms of the f_n 's are all bounded by some constant C , where $q > 2$.
3. The sequence f_n converges to 0 in the distributional sense.

then the $*$ -norm of the f_n 's will converge to zero as n goes to infinity. Such conditions are found to hold for a wide range of oscillating functions. For example in [43], it is shown that for the simple one-dimensional example $v(x) = \cos(xt)$, the $*$ -norm of v is $\frac{1}{t}$, and is thus inversely proportional to the oscillation in v .

An alternate to the $*$ -norm is found in the work by Yin et. al., which replaces the L^2 -norm of the residual in the second term of Equation 2.4.1 with the L^1 -norm (see [44]). It has been shown that the use of the L^1 -norm is especially effective in separating parts of an image into the cartoon or texture components based on their scale [44], creating a multiscale representation of an image. However, this so-called TV- L^1 model is not explored further in this thesis.

2.4.2 Vese-Osher Decomposition

The norm of Equation 2.4.2 for v motivated Vese and Osher in [2] to derive a practical implementation of the decomposition of f into the sum $u + v$, by approximating the L^∞ -norm of $\sqrt{g_1^2 + g_2^2}$ by the L^p -norm of the same quantity. Their energy functional is as follows

$$\begin{aligned} E_{\text{VO}}(u, g_1, g_2) = & \int_{\Omega} |\nabla u| dx dy + \lambda \int_{\Omega} (f - u - \partial_x g_1 - \partial_y g_2)^2 dx dy + \\ & \mu \left[\int_{\Omega} (\sqrt{g_1^2 + g_2^2})^p dx dy \right]^{\frac{1}{p}}. \end{aligned} \quad (2.4.4)$$

The model defined by the above energy is called the Vese-Osher (V-O) decomposition model. The values λ and μ are user-specified parameters, λ being a fidelity parameter controlling the amount of energy in the residual r , while μ controls how strongly the texture component v is modelled as having a small $*$ -norm, this norm being defined in Equation 2.4.2. The other variables and functions appearing in Equation 2.4.4 are as follows: f is the initial image, g_1 and g_2 are bounded functions, and Ω is the image space. The value p is the index of the L^p -norm, which is

set to 1 in [2] since this leads to simpler equations without substantial deterioration in image decomposition quality.

The first term of the V-O energy functional $E_{VO}(u, g_1, g_2)$ is a total variation term to ensure that the cartoon component u is piecewise smooth. The second term is a fidelity term to ensure that $f \approx u + v$. The last term is the L^p -norm approximation to the L^∞ -norm of $\sqrt{g_1^2 + g_2^2}$, which tries to make $\|v\|_*$ small, thus keeping the oscillating component of f in v .

If the V-O energy functional $E_{VO}(u, g_1, g_2)$ is formally minimized with the calculus of variations, the following system of Euler-Lagrange partial differential equations is obtained [2]

$$u = f - \partial_x g_1 - \partial_y g_2 + \frac{1}{2\lambda} \operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right) \quad (2.4.5)$$

$$\mu(\|\sqrt{g_1^2 + g_2^2}\|_p)^{1-p} (\sqrt{g_1^2 + g_2^2})^{p-2} g_1 = 2\lambda \left[\partial_x u - \partial_x f + \partial_{xx}^2 g_1 + \partial_{xy}^2 g_2 \right] \quad (2.4.6)$$

$$\mu(\|\sqrt{g_1^2 + g_2^2}\|_p)^{1-p} (\sqrt{g_1^2 + g_2^2})^{p-2} g_2 = 2\lambda \left[\partial_y u - \partial_y f + \partial_{xy}^2 g_1 + \partial_{yy}^2 g_2 \right]. \quad (2.4.7)$$

Here $\|\sqrt{g_1^2 + g_2^2}\|_p = (\int_{\Omega} (\sqrt{g_1^2 + g_2^2})^p dx dy)^{\frac{1}{p}}$, the L^p -norm of $\sqrt{g_1^2 + g_2^2}$. Vese and Osher's (u, v) decomposition is implemented in their paper with a semi-implicit fixed-point iterative finite differences scheme, based on that of Aubert and Vese [45].

The final equations that are iterated are as follows

$$u_{i,j}^{n+1} = \left(\frac{1}{1 + \frac{1}{2\lambda}(c_1 + c_2 + c_3 + c_4)} \right) \left[f_{i,j} - \frac{g_{1,i+1,j}^n - g_{1,i-1,j}^n}{2} - \frac{g_{i,j+1}^n - g_{i,j-1}^n}{2} + \frac{1}{2\lambda}(c_1 u_{i+1,j}^n + c_2 u_{i-1,j}^n + c_3 u_{i,j+1}^n + c_4 u_{i,j-1}^n) \right] \quad (2.4.8)$$

$$g_{1,i,j}^{n+1} = \left(\frac{2\lambda}{\mu H(g_{1,i,j}^n, g_{2,i,j}^n) + 4\lambda} \right) \left[\frac{u_{i+1,j}^n - u_{i-1,j}^n}{2} - \frac{f_{i+1,j} - f_{i-1,j}}{2} + g_{i+1,j}^n + g_{1,i-1,j}^n + \frac{1}{2}(2g_{2,i,j}^n + g_{2,i-1,j-1}^n + g_{i+1,j+1}^n - g_{2,i,j-1}^n - g_{i-1,j}^n - g_{2,i+1,j}^n - g_{2,i,j+1}^n) \right] \quad (2.4.9)$$

$$g_{2,i,j}^{n+1} = \left(\frac{2\lambda}{\mu H(g_{1,i,j}^n, g_{2,i,j}^n) + 4\lambda} \right) \left[\frac{u_{i,j+1}^n - u_{i,j-1}^n}{2} - \frac{f_{i,j+1} - f_{i,j-1}}{2} + g_{2,i,j+1}^n + g_{2,i,j-1}^n + \frac{1}{2}(2g_{1,i,j}^n + g_{1,i-1,j-1}^n + g_{1,i+1,j+1}^n - g_{1,i,j-1}^n - g_{1,i-1,j}^n - g_{1,i+1,j}^n - g_{1,i,j+1}^n) \right]. \quad (2.4.10)$$

The function $H(g_1, g_2)$ in Equations 2.4.9 and 2.4.10 is defined as $H(g_1, g_2) = (\|\sqrt{g_1^2 + g_2^2}\|_p)^{1-p}(\sqrt{g_1^2 + g_2^2})^{p-2}$. The constants $\{c_i\}_{i=1}^4$, dependent on u , are given

by the expressions

$$\begin{aligned}
c_1 &= \frac{1}{\sqrt{\left(u_{i+1,j}^n - u_{i,j}^n\right)^2 + \left(\frac{u_{i,j+1}^n - u_{i,j-1}^n}{2}\right)^2}}, \\
c_2 &= \frac{1}{\sqrt{\left(u_{i,j}^n - u_{i-1,j}^n\right)^2 + \left(\frac{u_{i-1,j+1}^n - u_{i-1,j-1}^n}{2}\right)^2}}, \\
c_3 &= \frac{1}{\sqrt{\left(\frac{u_{i+1,j}^n - u_{i-1,j}^n}{2}\right)^2 + \left(u_{i,j+1}^n - u_{i,j}^n\right)^2}}, \\
c_4 &= \frac{1}{\sqrt{\left(\frac{u_{i+1,j-1}^n - u_{i-1,j-1}^n}{2}\right)^2 + \left(u_{i,j}^n - u_{i,j-1}^n\right)^2}}.
\end{aligned}$$

2.4.3 Osher-Solé-Vese Decomposition

Another attempt at minimizing the $*$ -norm of Equation 2.4.2 was made by Osher, Solé and Vese in [4]. They used the Hodge decomposition of \vec{g} , which splits \vec{g} into the sum of the divergence of a single valued function and a divergence-free vector field. The Osher-Solé-Vese (OSV) model only consists of two components, the cartoon component u and the texture/noise component $v = f - u$, with f the original image. In this model, using the Hodge decomposition of \vec{g} and ignoring the divergence-free field part, it is found that the texture component v should have a small norm in the Sobolev space $H^{-1}(\Omega)$, where this norm is defined as

$$||v||_{H^{-1}(\Omega)}^2 = \int_{\Omega} |\nabla(\Delta^{-1})v|^2 dx dy.$$

The space $H^{-1}(\Omega)$ is not explicitly or directly used in the derivation leading up to the fact that the norm of v in that space should be small, however the final integral term $\int_{\Omega} |\nabla(\Delta^{-1})v|^2 dx dy$ which is found to model the texture when small, happens to coincide with the definition of the norm on this space. Note that in this term $(\Delta^{-1})v$ refers to the function h such that $\Delta h = v$, or in other words, satisfies the Poisson equation with right-hand side v .

Thus, Osher, Solé and Vese obtained the following energy to be minimized by ignoring the divergence-free part of \vec{g}

$$\mathbf{E}_{\text{OSV}}(u) = \int_{\Omega} |\nabla u|^2 dx dy + \lambda \int_{\Omega} |\nabla(\Delta^{-1})(f - u)|^2 dx dy \quad (2.4.11)$$

In [4], a PDE that is a gradient descent solution for the above energy is found to be

$$u_t = \frac{1}{2\lambda} \Delta \left[\text{div} \left(\frac{\nabla u}{|\nabla u|} \right) \right] - (u - f) \quad (2.4.12)$$

with adiabatic boundary conditions, meaning that the boundary of u is padded with repetition at each iteration. The resulting energy functional contains an inverse Laplacian of $f - u$, but this was eliminated by showing that under some rather relaxed conditions, the equation $u_t = \Delta \mathbf{E}'(u)$ always decreases the energy $\mathbf{E}_{\text{OSV}}(u)$, or in other words, is a gradient descent direction of that energy, just like the usual gradient descent equation $u_t = -\mathbf{E}'(u)$. The auxiliary functions g_1 and g_2 are no longer involved in the PDE, as they disappear in the derivation. In this thesis, the Osher, Solé and Vese decomposition model is called the OSV model, after the authors. In Chapter 5, this model is adapted and extended, first by adding an extra term for decorrelation of the cartoon and texture components, and then

modifying the Laplacian by introducing: a) a conductance coefficient inside of it, and (b) using an oriented Laplacian instead, both in attempts to make the model more effective for denoising.

In those functionals that include g_1 and g_2 , e.g. that of Vese and Osher [2] described in Section 2.4.2, the functions g_1 and g_2 can further be used for texture segmentation/discrimination by applying an Active Contour without Edges model, as found in [3], on $|g_1|$ or $|g_2|$. This procedure can be time consuming, as demonstrated by our experiments. A simpler and more efficient method based on the periodic texton model of textures [46] can be used by measuring the inhomogeneity of a blurred version of the square of one of the subcomponents g_1 or g_2 , if the blurring is done with a Gaussian of appropriate variance. This type of efficient method is used in Chapters 3 and 4. Since the OSV model does not include g_1 and g_2 , it cannot be directly used in the fashion described for texture segmentation/discrimination.

Level set methods are now described, which are a vital part of the Active Contour without Edges (ACWE) model. The ACWE model is in turn applied to discrimination of images based on subcomponents from their decomposition.

2.5 Level Set Methods

Decomposition is the algorithm that feeds into texture discrimination in Vese and Osher's paper [2]. After a brief introduction to level set methods in this subsection, the Active Contours without Edges scheme, which is based on level sets, is examined in the following subsection.

Once an image has been decomposed, it is possible to use the subcomponents of the texture component v (g_1 and g_2) for texture segmentation/discrimination, as was done in [2], and as was explained at the end of the previous section. Active Contours Without Edges, which has a natural implementation with level set methods, is a way to perform this segmentation/discrimination.

Since their invention by Osher and Sethian [47], level set methods have found widespread use in the fields of image processing and computer vision. For the problem of segmentation, a curve is overlaid on the image to be segmented, and the curve is allowed to evolve until it “hugs” the boundary of the object(s).

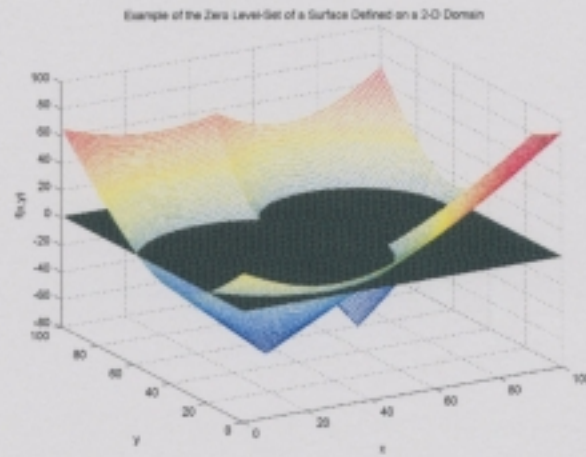
The key observation behind level set methods is that a curve in 2-D space can be regarded as being the zero level-set of a three-dimensional surface (see Figure 2.4). The zero level-set is defined as being the intersection of the surface $z = f(x, y)$ with $z = 0$. It can be shown (see [48]) that if a curve C evolves over an artificial time t according to the partial differential equation

$$\frac{\partial C}{\partial t} = \beta \vec{N}, \quad (2.5.1)$$

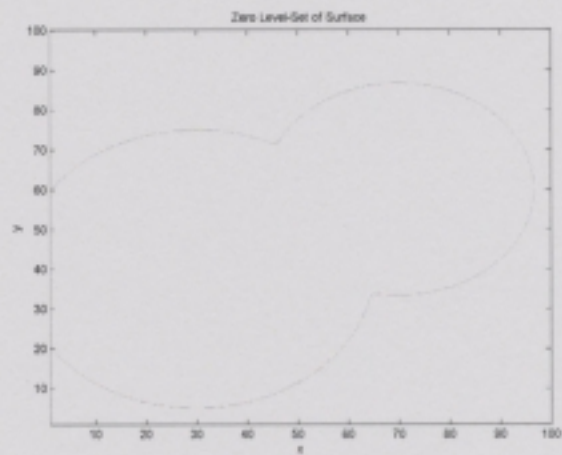
where C and \vec{N} both depend on time and a parameterized arclength (\vec{N} is the normal vector to the curve C at any point), and β depends on the curvature of C , then the corresponding level set function ϕ will evolve according to the equation

$$\frac{\partial \phi}{\partial t} = \beta |\nabla \phi|. \quad (2.5.2)$$

This equation is usually solved iteratively via forward time-stepping with a time increment Δt , i.e.



(a)



(b)

Figure 2.4: Formation of zero level-set by taking level cross-section of a 3-D surface

$$\phi_{i,j}^{n+1} = \phi_{i,j}^n + \Delta t \beta_{i,j}^n |\nabla \phi_{i,j}^n|$$

A common procedure that must be incorporated into many practical level set methods is reinitialization. This specifically means that after every N_r iterations of the main level set method (with e.g. $N_r = 20$), the level set function z must be reinitialized to a signed distance function to avoid the gradient of z from becoming too large or small [29]. If this were to happen, then the curvature of the level set function would become undefined, and spikes would appear in z . The PDE that is used for reinitialization is $z_t = \text{sgn}(z)(1 - |\nabla z|)$ [29]. This imposes the condition that the norm of the gradient z is equal to 1, which means that indeed z will be a signed distance function.

2.6 Active Contours without Edges

Active Contours Without Edges (ACWE) [3] is an active contours method, which means it involves an evolving curve or curves. A simple example of an active contour method is mean curvature motion [49], where an evolving curve C travels at a speed equal to the curvature of C normal to the curve. become convex and converge to a circle [48]. Active contour models are used primarily for image segmentation so that the curve reaches equilibrium once it “hugs” the edges of an object in the image.

Most active contour models developed thus far have been gradient based (e.g. [50], [51]), which means that the evolving curve’s speed is dependent on the gra-

dient of the image over which the curve is evolving. Thus these image gradients must be explicitly calculated making these methods sensitive to noise. What distinguishes the ACWE model from more traditional models is that no such gradient calculations are needed. Instead the energy functional for the curve tries to minimize the sum of the variances of the image inside and outside of the curve. Also, there is a term in the functional for the curve length, so that the curve which defines the boundary between two objects is itself smooth. The Active Contours Without Edges model is implemented with the level set framework; generalizations of the model have been extended to the segmentation of more than two regions [52] (by using more than one level set function), as well as to vector-valued images [53].

The energy functional that is to be minimized for the ACWE model is

$$\begin{aligned} F(m_1, m_2, C) = & \mu_{ACWE} \cdot Length(C) + \lambda_1 \int_{inside(C)} (u_0(x, y) - m_1)^2 dx dy + \\ & \lambda_2 \int_{outside(C)} (u_0(x, y) - m_2)^2 dx dy, \end{aligned} \quad (2.6.1)$$

where u_0 is the image being segmented, C is the curve which is segmenting the image, and m_1 and m_2 are two constants which vary with each iteration and are calculated to be the average of u_0 inside and outside of the curve C respectively. λ_1 and λ_2 are both weighting parameters, which are set to 1, as is done in [3]. The only other parameter is μ_{ACWE} , which weights the length of the contour. In [3], it is demonstrated via examples that the larger μ_{ACWE} , the more likely that large objects will be detected and that smaller objects will be grouped together. The algorithm is more robust to noise with large values of μ_{ACWE} .

This energy functional is more easily solved using a level-set formulation. As-



Figure 2.5: Sample Initialization of Level Set Function ϕ overlaid on $|g_1|$ subcomponent of Image with Two Separate Textures

suming that the curve C which forms the boundary between the two regions of the image is the zero level-set of the function ϕ , the energy which must be minimized by ϕ is

$$\begin{aligned} E_{ACWE}(\phi) = & \mu_{ACWE} \int_{\Omega} \delta(\phi(x, y)) |\nabla \phi(x, y)| dx dy \\ & \lambda_1 \int_{\Omega} (u_0(x, y) - m_1)^2 H(\phi(x, y)) dx dy \\ & + \lambda_2 \int_{\Omega} (u_0(x, y) - m_2)^2 (1 - H(\phi(x, y))) dx dy. \end{aligned} \quad (2.6.2)$$

where $H(\cdot)$ is the Heaviside or step function. The Euler-Lagrange equation for the given functional is

$$\frac{\partial \phi}{\partial t} = \delta_{\epsilon}(\phi) \left[\mu \operatorname{div} \left(\frac{\nabla \phi}{|\nabla \phi|} \right) - \lambda_1 (u_0 - m_1)^2 + \lambda_2 (u_0 - m_2)^2 \right]. \quad (2.6.3)$$

with adiabatic boundary conditions and with the initial condition $\phi(x, y, t = 0) = \phi_0(x, y)$. In practice, $\phi_0(x, y)$ is set to be a series of small vertically and horizontally aligned circles, as this speeds up convergence of ACWE in Equation 2.6.3. An example of such an initialization is given in Figure 2.5.

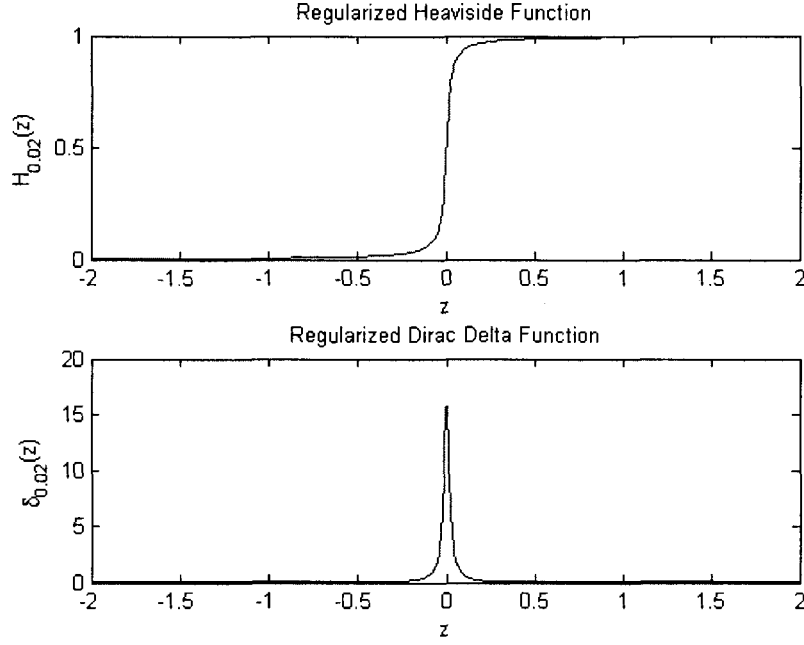


Figure 2.6: $H_{0.02}(z)$ and $\delta_{0.02}(z)$

A regularized delta function δ_ϵ is used above and is found by taking the derivative of the regularized Heaviside function [3]. The regularized Heaviside function, $H_\epsilon(z)$, that is chosen for the purposes of this thesis, is given by the expression

$$H_\epsilon(z) = \frac{1}{2} \left(1 + \frac{2}{\pi} \arctan \left(\frac{z}{\epsilon} \right) \right), \quad (2.6.4)$$

and the corresponding regularized delta function $\delta_\epsilon(z)$ is

$$\delta_\epsilon(z) = \frac{1}{\pi} \frac{\epsilon}{\epsilon^2 + z^2} \quad (2.6.5)$$

Plots of these two functions, with $\epsilon = 0.02$ are shown in Figure 2.6.

2.7 Orientation and Coherence Calculation

The algorithms presented in Chapters 5 and 6 require the noise-resistant computation of orientation at each pixel in the image in addition to the detection of which image regions are coherent, that is roughly consisting of one orientation. There are two methods from the literature which are considered here, in which such a calculation robust to noise can be performed, both of which use the structure tensor, or the outer product of the image gradient with itself. The first is based on the linear structure tensor, which corresponds to a linear diffusion or Gaussian blurring of the structure tensor elements, and the second is based on the nonlinear structure tensor, an extension of the linear structure tensor which corresponds to nonlinear diffusion of the structure tensor entries. Both of these orientation calculation methods are now described in the next section, along with methods of determining the orientation coherence. If not otherwise specified in the following text, it is assumed the orientation refers to the gradient orientation, or the orientation in which the image changes the most in intensity.

2.7.1 Linear and Nonlinear Structure Tensors

The structure tensor is defined as the outer product of the image gradient vector with itself [31]. Supposing that the image is f , then the structure tensor J is defined as

$$J = \begin{pmatrix} f_x^2 & f_x f_y \\ f_x f_y & f_y^2 \end{pmatrix}.$$

The structure tensor is blurred elementwise with a Gaussian filter of standard

deviation σ , to obtain $J_\sigma = G_\sigma * J$. Robustness to noise is obtained because of this blurring operation. Then the orientation at each pixel is computed as the eigenvector \vec{w}_1 of J_σ corresponding to its larger eigenvalue λ_1 . This corresponds to the gradient direction at each pixel, and the isophote direction is simply computed as the eigenvector corresponding to the smaller eigenvalue of the structure tensor, since J_σ can be shown to have orthogonal eigenvectors.

The nonlinear structure tensor [54] is an extension of the linear structure tensor to compute the orientation field of an image f . The advantages of this nonlinear tensor are lack of blurring of adjacent different orientations and resistance to noise. The nonlinear structure tensor is obtained from nonlinear matrix diffusion of the structure tensor, which is found by the outer product of the image gradient vector with itself, as just described. Nonlinear diffusion using Additive Operator Splitting is done separately on each element of the matrix, but with a diffusivity function g (with scalar argument) jointly computed from all the matrix entries

$$\partial_t J^{ij} = \text{div} \left(c \left(\sum_{k,l=1}^2 |\nabla J^{kl}|^2 \right) \nabla J^{ij} \right).$$

The diffusivity (conductivity) function is chosen as $c(|\nabla u|^2) = \frac{1}{|\nabla u|}$, corresponding to total variation flow. Then, the gradient and isophote directions at each pixel are computed in a manner similar to the linear structure tensor above.

It was found that though the orientations determined by the nonlinear structure tensor were sometimes more accurate than those from the linear structure tensor, for the decomposition models in this thesis, the linear structure tensor estimates were sufficient. Thus, these estimates from the linear structure tensor were used,

since they were much more efficient in their computation, not requiring an iterative diffusion process as for the nonlinear structure tensor.

2.7.2 Orientation Coherence

In Chapters 5 and 6, it is necessary to determine which regions are oriented and which are not, so that different variational models can be applied to each type of region. A region is defined to be non-oriented when its gradient direction coherence [55] is less than a pre-determined threshold, and this coherence function is a measure of how uniform are gradient directions around a pixel. In [56], the coherence of f is measured directly using a small window W around each pixel by the formula

$$coher(\theta_{i,j}) = |\nabla f|_{i,j} \frac{\sum_{(u,v) \in W} ||\nabla f|_{u,v} \cos(\theta_{i,j} - \theta_{u,v})|}{\sum_{(u,v) \in W} |\nabla f|_{u,v}},$$

where $\theta_{i,j}$ is the orientation calculated from the linear structure tensor at pixel (i, j) ; generally W is chosen to be 7 pixels by 7 pixels. However, the coherence can also be directly computed from the structure tensor as the difference between the two eigenvalues of the evolved structure tensor squared, i.e. $(\lambda_1 - \lambda_2)^2$. There are other expressions for the gradient direction coherence, e.g. $\left(\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}\right)^2$. The latter estimate has the advantage of always lying between 0 and 1. These coherence estimates obtained directly from the eigenvalues of the structure tensor were not generally used in this thesis.

2.8 Chapter Summary

In this chapter, the Energy Method was described, which is one new approach to solving image processing problems in the literature. Image decomposition was explained in more depth than in the first chapter, with examples of existing decomposition models given. In addition to this, major mathematical and image processing auxiliary methods, such as Additive Operator Splitting and Active Contours Without Edges, that are utilized by the models in the remainder of the thesis were also presented.

In the thesis roadmap in Section 1.8, brief descriptions of the various new decomposition models were given. These new models are Improved Edge Segregation (IES, Chapter 3), Simultaneous Decomposition/Discrimination (SDD, Chapter 4), Decorrelated Osher-Solé-Vese Decomposition (DOSV, Chapter 5), Oriented Laplacian Osher-Solé-Vese Decomposition (OLOSV, Chapter 5), Orientation-Adaptive Decomposition (OAD, Chapter 6) and Eikonal Orientation-Adaptive Decomposition (EOAD, Chapter 6). Table 2.1 lists each tool/model that was exposted in this chapter, alongside the subsection in which the tool/model was introduced, and checkmarks indicate which of the proposed new image decomposition models use the tool or model.

Section	Tool	Decomposition Model					
		IES	SDD	DOSV	OLOSV	OAD	EOAD
2.2	AOS	✓	✓				
2.3	V-O Decomposition	✓	✓			✓	✓
2.3	OSV Decomposition			✓	✓		
2.4	Level Set Methods		✓				
2.5	ACWE		✓				
2.6	Orientation Coherence				✓	✓	✓

Table 2.1: List of Tools/Models and the Decomposition Models Proposed in this Thesis which Use them

CHAPTER 3

Improved Edge Segregation

3.1 Introduction

As mentioned in the previous chapter, the decomposition model of Vese and Osher [2] was the first practical implementation of the variational decomposition framework of Meyer [1]. In their paper, Vese and Osher iteratively solved the Euler-Lagrange equations from their model with Gauss-Seidel iterations (see Section 2.2.1.2). This leads to a slow energy minimization procedure. Also, in the experiments conducted in the research of this thesis, it was found that sometimes, cartoon edges would appear quite strongly in the texture component of the decomposition. This should not happen, because such edge information should appear as much as possible in the cartoon component, and not in the texture component where it does not belong. These two issues are addressed by proposing new decomposition models in the following sections, of which one, the Improved Edge Segregation model, is particularly successful in giving good decomposition results in a short amount of time.



(a) Original barbara image

(b) u component using Jacobi iteration



(c) u component using Gauss-Seidel iteration

Figure 3.1: Original barbara image and u component using Jacobi and Gauss-Seidel iteration implementations of the Vese-Osher model for image decomposition of barbara (50 iterations, $\mu = 0.05$, $\lambda = 0.05$)

3.2 Example of Instability

Figure 3.1 shows what occurs to the u (cartoon) component when an explicit Jacobi method is used to discretize the Euler-Lagrange equations in the image decomposition scheme of Vese and Osher [2]. There is clearly some instability, as seen in the large amplitude oscillations in Figure 3.1(b). This is not seen with the Gauss-Seidel fixed-point iteration used in [2], depicted in Figure 3.1(c).

In this chapter, three possible sets of terms are proposed to be added to Vese and Osher's energy functional. One of these sets of terms does not split individual edges between the cartoon and textured components (Edge Segregation), another imposes a geometric constraint, and another is a better version of Edge Segregation called Improved Edge Segregation. All these terms require a measure of texture inhomogeneity; two of which defined in subsequent sections are $|\nabla \vec{g}|$ (defined in Section 3.3) and $|\tilde{\nabla} \vec{g}|$ (defined in Section 3.4).

The aim is to use the additional terms to provide regularization of the Vese-Osher functional by imposing *a priori* conditions on the output decomposition, to the point that Jacobi iteration can be used instead of the usual Gauss-Seidel for Vese-Osher. As indicated above, this is very desirable since a Jacobi iteration implementation is more parallelizable than a Gauss-Seidel one, and thus can be made much more efficient than a Gauss-Seidel method, despite the additional complexity of the Euler-Lagrange equations of the new methods themselves (due to the extra terms). These three sets of terms that were added to the Vese-Osher functional are now discussed.

3.3 Edge Segregation

Recall from Equation 2.4.4 that Vese and Osher's energy functional is given by the expression

$$\mathbf{E}_{\mathbf{VO}}(u, g_1, g_2) = \int_{\Omega} |\nabla u| dx dy + \lambda \int_{\Omega} (f - u - \partial_x g_1 - \partial_y g_2)^2 dx dy + \mu \left[\int_{\Omega} (\sqrt{g_1^2 + g_2^2})^p dx dy \right]^{\frac{1}{p}} \quad (3.3.1)$$

A new energy $\mathbf{E}_{\mathbf{ES}}$ is defined by adding a term to $\mathbf{E}_{\mathbf{VO}}$. This new energy is

$$\mathbf{E}_{\mathbf{ES}}(u, g_1, g_2) = \mathbf{E}_{\mathbf{VO}}(u, g_1, g_2) + \gamma \int_{\Omega} |\nabla u| (G_{\sigma} * |\nabla \vec{g}|) dx dy, \quad (3.3.2)$$

where

$$|\nabla \vec{g}| = \sqrt{g_{1,x}^2 + g_{1,y}^2 + g_{2,x}^2 + g_{2,y}^2}. \quad (3.3.3)$$

The coefficient γ is a real-valued regularization parameter, G_{σ} is a Gaussian filter of standard deviation σ , and $*$ denotes convolution. This new term attempts to ensure that edges are either kept in the u component or the v component but not both, because the energy becomes high when edge strengths in both the u and v components are high. This follows from an approximation from the arithmetic-geometric mean inequality [57].

Because $f \approx u + v$, this means that $\nabla f \approx \nabla u + \nabla v$, assuming that the gradient

operator does not amplify the residual $r = f - u - v$ too greatly. By definition,

$$\begin{aligned}
v &= g_{1,x} + g_{2,y} \\
\Rightarrow \nabla v &= (v_x, v_y) = (g_{1,xx} + g_{2,xy}, g_{1,xy} + g_{2,yy}) \\
\Rightarrow |\nabla v| &= \sqrt{(g_{1,xx} + g_{2,xy})^2 + (g_{1,xy} + g_{2,yy})^2}.
\end{aligned} \tag{3.3.4}$$

For ease of implementation, the approximation

$$|\nabla v| = G_\sigma * |\nabla \vec{g}| \tag{3.3.5}$$

is used, where $|\nabla \vec{g}|$ is given by the expression in Equation 3.3.3, G_σ is a Gaussian filter of standard deviation σ and $*$ is the convolution operation. Using the exact form of ∇v was found experimentally to cause amplification of noise in the evolution of image decomposition, which led to instability in the early stages of the fixed point iteration.

3.3.1 Justification of Approximation

Next, we turn to the justification of this approximation in Equation 3.3.5. Because $f \approx u + v$, $\nabla f \approx \nabla u + \nabla v$, and this approximation was found through tests to be especially true away from very small-scale texture, since then the gradient operator did not unduly amplify the residual component. The error in this approximation is manifested in the sometimes relatively poor texture quality measures obtained in Section 3.7.6.1, where a possible remedy is proposed. Assuming that the approximation is valid, then by definition, $v = g_{1,x} + g_{2,y}$ which implies that

$$\nabla v = (v_x, v_y) = (g_{1,xx} + g_{2,xy}, g_{1,xy} + g_{2,yy}).$$

As in Equation 3.3.4, $\nabla v = (g_{1,xx} + g_{2,xy}, g_{1,xy} + g_{2,yy})$. In Equations A.1.4 and A.1.5 of Appendix A, it is shown that $g_1 = \frac{1}{2\mu} \frac{\partial}{\partial x} K(u)$ and $g_2 = \frac{1}{2\mu} \frac{\partial}{\partial y} K(u)$, where $K(u)$ is the curvature of the level lines of the cartoon component u . Therefore,

$$g_{1,y} = g_{2,x} = \frac{1}{2\mu} \frac{\partial^2}{\partial x \partial y} K(u).$$

Admittedly, this holds for the case of minimizing $\|v\|_*^2$, where the exponent of the approximating L^p -norm is taken to be $p = 2$, since this is the assumption from [4]. However, this is considered to be a good approximation to the solution of the IES model, since both models approximate the L^∞ -norm in the $*$ -norm from the basic model of Meyer of Equation 2.4.2, and so the derivation and justification of the new terms assume this relation is true.

Because $g_{1,y} = g_{2,x}$, and $v = g_{1,x} + g_{2,y}$, we obtain

$$v_x = g_{1,xx} + g_{2,xy} = g_{1,xx} + g_{1,yy} = \Delta g_1, \quad (3.3.6)$$

where Δ is the Laplacian operator. Then, if a Fourier Series approximation of each side of this equation is used, as in [4], it can be shown that g_1 will be a negative and attenuated version of v_x . Thus, $g_{1,x}$ will be a negative and attenuated version of v_{xx} . In a similar fashion, $g_{2,y}$ will be a negative and attenuated version of v_{yy} .

It is well known from the image processing literature [18] that the second derivative of a signal will produce spikes adjacent to and on both sides of an edge. This is the basis for the well known Laplacian zero crossings technique for edge detection.

When g_1 or g_2 is blurred with a Gaussian of small standard deviation ($\sigma = 1$ or 2), as is done in the calculation of $|\nabla \vec{g}|$, a large value of this gradient at *and* very close to the edges of the texture component v is obtained. This approximation was found to be much more stable than the direct $|\nabla v|$ quantity of Equation 3.3.4, and it is also more efficient.

3.3.2 Euler-Lagrange Equations

The additional edge segregation term in Equation 3.3.2 changes the Euler-Lagrange equations for u , g_1 and g_2 from those in Equations 2.4.5-2.4.7 to the following

$$u = f - \partial_x g_1 - \partial_y g_2 + \frac{1}{2\lambda} \operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right) + \frac{\gamma}{2\lambda} \operatorname{div} \left(\frac{(G_\sigma * |\nabla \vec{g}|) \nabla u}{|\nabla u|} \right) \quad (3.3.7)$$

$$\mu \frac{g_1}{\sqrt{g_1^2 + g_2^2}} = 2\lambda [u_x - f_x + \partial_{xx}^2 g_1 + \partial_{xy}^2 g_2] + \gamma \operatorname{div} \left(\frac{|\nabla u| (G_\sigma * \nabla g_1)}{G_\sigma * |\nabla \vec{g}|} \right) \quad (3.3.8)$$

$$\mu \frac{g_2}{\sqrt{g_1^2 + g_2^2}} = 2\lambda [u_y - f_y + \partial_{xy}^2 g_1 + \partial_{yy}^2 g_2] + \gamma \operatorname{div} \left(\frac{|\nabla u| (G_\sigma * \nabla g_2)}{G_\sigma * |\nabla \vec{g}|} \right) \quad (3.3.9)$$

The resulting Euler-Lagrange equations were discretized using a semi-implicit scheme without a clear-cut time step, the time step being understood to be equal to 1. This scheme is based on that of Vese and Osher, but does not use the most recently calculated function values at an image point, instead always using the values from the previous time step. In other words, here a Jacobi method is used, whereas as described in Section 3.2, Vese and Osher must use a Gauss-Seidel method to force convergence and ensure stability. For this scheme, the PDEs are discretized by keeping them in divergence form and enforcing Dirichlet boundary conditions

(zero-padding) on g_1 and g_2 and Neumann (adiabatic or repeating) boundary conditions on u (see [2] for more details). Due to the increased stability of the proposed scheme, and in addition to this benefit, due to increased efficiency, the computations are vectorized. This leads to a time savings, offsetting the additional complexity of the Euler-Lagrange equations themselves.

3.3.3 Residual with Edge Segregation

Unfortunately, despite the increased stability of this model, and the ability to discretize it using Jacobi iterations, the model itself is flawed. It was found that there was nothing to prevent cartoon edges and other image information from being transferred to the residual r , because both $|\nabla u|$ and $G_\sigma * |\nabla \vec{g}|$ can be low at edges in f , where one of them should be high. In fact, this does happen in practice. In Section 3.5, a more elaborate model is proposed to circumvent this problem.

An example of the increased amplitude of the residual r component (with 130 added to its graylevel for visualization purposes) is shown in Figure 3.2 with the test image barbara from Figure 3.1(a). The parameters for Vese-Osher (V-O) decomposition were $\lambda = \mu = 0.05$, and this method was run for 50 iterations. The parameters for Edge Segregation that were used here were $\lambda = \mu = 0.05$ and $\gamma = 0.2$, though any positive value of γ , where the Edge Segregation term is included would lead to a residual component which is too great and which should include extra information which should instead be present in the cartoon and texture components.

This method was also run for 50 iterations. It is seen that the residual r compo-

nent is too strong with Edge Segregation which means that cartoon/texture information is lost from the u and v components.

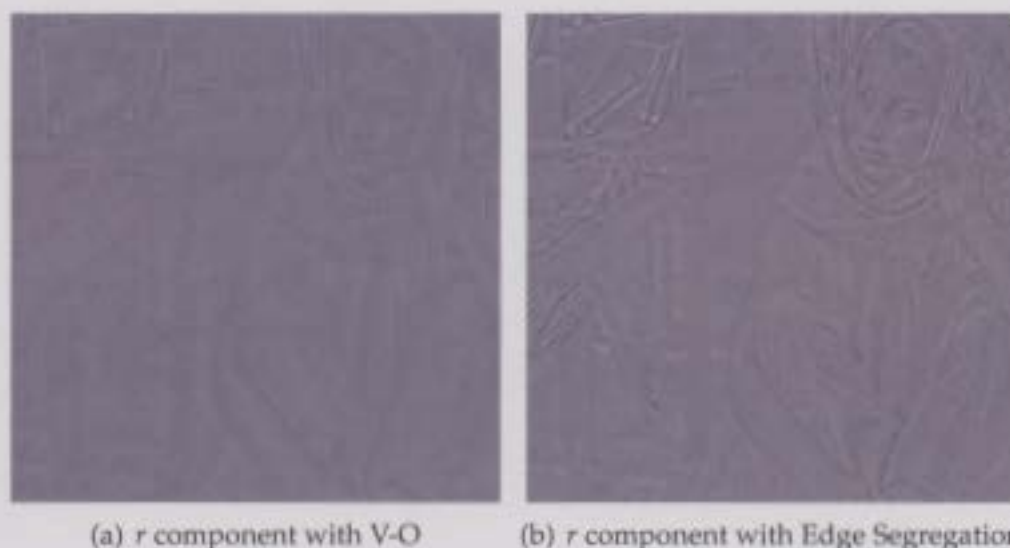


Figure 3.2: Comparison of residual r component between V-O and Edge Segregation Methods

3.4 Geometric Constraint

In this section, a geometric constraint between the texture and cartoon components of the V-O image decomposition model is introduced. Recently (Nov. 2004), Aujol and Chan [58] used a logical framework to combine non-texture and texture channels. However this was for supervised classification, where each pixel in the image is assigned to one of several given classes. In the work leading to this thesis, it was observed that there are natural constraints on the textured (v) component of an image. Each texture in this component should lie completely within one region of the cartoon or u component. This could require the ability to measure the gradient of the texture; as explained later, this does not have to be done using the model

of [2]. Then this measurement of texture boundaries is applied to other proposed decomposition models. These ideas are now developed.

To impose a geometric constraint, the following term was added to the Vese-Osher energy functional for decomposition of textured images

$$\gamma GCT(u, g_1, g_2) = \gamma \int_{\Omega} \frac{|\nabla u|}{|\tilde{\nabla} \vec{g}_{init}| + \epsilon} dx dy. \quad (3.4.1)$$

Here γ is a weighting parameter, and ϵ is a small positive constant which is set to 1 to ensure that the denominator does not vanish. Thus, an energy $E_{GC}(u, g_1, g_2)$ can be defined as

$$E_{GC}(u, g_1, g_2) = E_{VO}(u, g_1, g_2) + \gamma GCT(u, g_1, g_2) \quad (3.4.2)$$

Prior to defining the expression $|\tilde{\nabla} \vec{g}_{init}|$ in Equation 3.4.1, consider first defining a quantity $|\tilde{\nabla} \vec{g}|$ as

$$|\tilde{\nabla} \vec{g}| = \sqrt{|(G_{\sigma} * (g_1^2))_x| + |(G_{\sigma} * (g_1^2))_y| + |(G_{\sigma} * (g_2^2))_x| + |(G_{\sigma} * (g_2^2))_y|}. \quad (3.4.3)$$

Here, $|\tilde{\nabla} \vec{g}|$ is a measure of texture heterogeneity — the higher its value at a pixel, the more likely that there is a texture boundary at that pixel.

Then, we can let $|\tilde{\nabla} \vec{g}_{init}|$ refer to the value of $|\tilde{\nabla} \vec{g}|$ after a small number N_g of iterations. Unless stated otherwise, N_g was chosen to be 10, since this was sufficiently small for efficiency purposes, and enough information had been transferred to g after this number of iterations. $|\tilde{\nabla} \vec{g}_{init}|$ is calculated by setting $\gamma = 0$ for the first N_g iterations, so that for this number of iterations, Vese and Osher's origi-

nal model is used, and not the proposed model (because the energy functional $E_{GC}(u, g_1, g_2) = E_{VO}(u, g_1, g_2) + \gamma GCT(u, g_1, g_2)$). After this, γ can be set to any constant value. We now briefly discuss why there is a Gaussian blurring operator in the definition of Equation 3.4.3.

3.4.1 Explanation of requirement to blur g_1 and g_2

An explanation of why Gaussian blurring is present in the expression for $|\tilde{\nabla} \vec{g}|$ in Equation 3.4.3 follows. From the relation $v = g_{1,x} + g_{2,y}$ in Equation 2.4.3, it can be determined, as done in Section 3.3.1, that g_1 and g_2 to an approximation have the same period as v (recall v is a texture, many of which display repetitions) in the x and y directions, respectively. This is because the period stays invariant with respect to derivatives. It is easy to show that g_1 is primarily large in absolute value at horizontal level regions of v , while g_2 is large in absolute value at vertical level regions. The intra-region means of g_1 and g_2 will be different. By blurring the squares of g_1 and g_2 with a Gaussian of the appropriate radius (approximately the same or greater than one period of v in each dimension), the result will be approximately constant within regions with a jump between different textures.

3.4.2 More on the Geometric Constraint Energy

The term in Equation 3.4.1 would penalize large cartoon gradients which occur in regions where the texture gradient is quite small. One way of viewing this is that the mean of a given texture generally does not change significantly with location. This is because the u cartoon component contains the structure of an image, and

if the means of the various image regions are associated with u , then the v component is zero-mean. For the vast majority of images, there are no cartoon edges within a given texture. An example of such an uncommon configuration of texture and cartoon components is shown below in Figure 3.3. A notable exception when such a configuration can occur is when one part of a texture region is illuminated by strong sunlight while the other is in the shade, an example of which is found in [59]. However, this is a very rare occurrence over the whole class of textured images.

The Euler-Lagrange equations for the new functional of Equation 3.4.2 contain the sgn function. These Euler-Lagrange equations are not included here since they do not add to the current discussion. The sgn function comes from differentiation of the absolute value function, which is found in the expression for $|\tilde{\nabla}g|$ in Equation 3.4.3.

It was found experimentally that the regularization power of such a geometric constraint was not strong enough to prevent instability of a Jacobi implementation when added to V-O decomposition. Regularization power refers to the ability of a modified model to restrict the class of possible minimizers of the Euler-Lagrange equation(s) of a system, and thus make the iterative process more stable. It is speculated that adding this term to an already constrained version of Vese-Osher decomposition, e.g. the Improved Edge Segregation model, would ameliorate results. This is an opportunity for further research, in addition to trying to implement it with Gauss-Seidel iterations, despite what are expected to be slower speeds than Jacobi iterations.

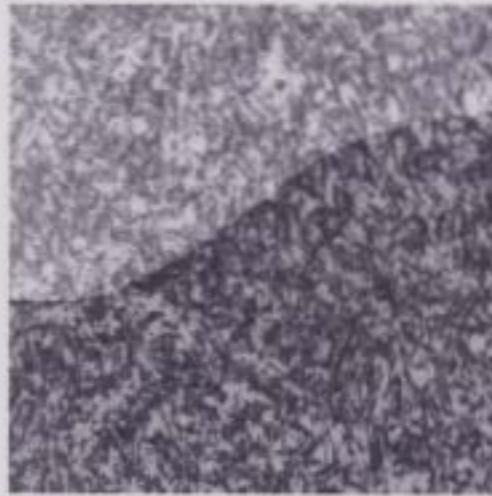


Figure 3.3: Artificial Image Violating the Geometric Constraint of Section 3.4

3.5 Improved Edge Segregation

3.5.1 Proposed New Terms

As already mentioned in Section 3.3.3, one problem found via runs of the Edge Segregation model on test images with the previously mentioned term in Equation 3.4.1 added to Vese and Osher's model is the fact that it allows for both the gradient of the u component and the v component to be small, so that much of the high gradient information can be moved to the residual r . To prevent this from happening, the addition of another two terms to the V-O image decomposition functional is proposed. The new energy, defining the Improved Edge Segregation (IES) decomposition model is

$$E_{IES}(u, g_1, g_2) = E_{VO}(u, g_1, g_2) - \gamma_1 \int_{\Omega} \frac{|\nabla u|}{G_{\sigma} * |\nabla \tilde{g}| + \epsilon_1} dx dy - \gamma_2 \int_{\Omega} \frac{G_{\sigma} * |\nabla \tilde{g}|}{|\nabla u| + \epsilon_2} dx dy \quad (3.5.1)$$

where γ_1 and γ_2 are both non-negative real coefficient parameters, which may depend on the image being decomposed and spatially within an image, and ϵ_1 and ϵ_2 are small positive constants introduced to avoid division by zero.

Though the V-O model was almost always unstable when implemented with Jacobi iterations, a wide range of choices of the coefficients γ_1 and γ_2 led to a stable Jacobi implementation of Improved Edge Segregation. The values of γ_1 and γ_2 chosen in the experiments in this chapter were image dependent, and were within this large range of stability (which was also image dependent). It is possible that different choices within these experimentally determined ranges could have led to even better decomposition results.

Observe that the coefficients (including the minus signs) of both new terms are negative. This means that the energy functional will favour these terms (other than the coefficients) to be positive and large, rather than small. Also note that the first subtracted term in Equation 3.5.1 is similar to the Geometric Constraint term in Equation 3.4.1.

The same justification for the use of $G_\sigma * |\nabla \vec{g}|$ in the above IES energy functional in Equation 3.5.1, as for the case of Edge Segregation in 3.3.1, can be made to show that this quantity is high at and close to textural edges.

The coefficients γ_1 and γ_2 of the second and third terms of Equation 3.5.1 in Section 3.5.1 are adapted in order that the regularization only occurs where it is needed. Both γ_1 and γ_2 depend on f_{NLD} , a nonlinear diffused version of the initial image f . Now as an aside, we describe this nonlinear diffusion flow both mathematically and through an illustrative example.

3.5.2 Example of Nonlinear Diffusion Flow

The whole purpose of nonlinear diffusion flow is to circumvent the blurring of edges which occurs with usual Gaussian blurring (e.g. see Section 2.2.1.3). An Additive Operator Splitting (AOS) isotropic diffusion scheme [36], as described in Section 2.3, was used for the implementation of nonlinear diffusion flow in IES. A fidelity term ($\int_{\Omega} (f - u)^2 dx dy$) was not used in the nonlinear diffusion functional, though one could be used, at the expense of having to choose another weighting parameter. The general form of such nonlinear diffusion is $\partial_t u = \text{div}(g(|\nabla u|^2) \nabla u)$. For the total variation flow already described in Section 2.4.1, $g(|\nabla u|^2) = \frac{1}{|\nabla u|}$. It was found from experiments that the TV flow still blurred some edges too much, so a weaker diffusivity function, $g(|\nabla u|^2) = \frac{1}{|\nabla u|^2}$, was used instead.

As described in Section 2.3, AOS is characterized by a semi-implicit discretization in time with computational steps and memory requirements linear in the number of pixels of the image on which AOS is performed. The Thomas algorithm, described in [36], is used to solve a tridiagonal linear system and AOS also involves the use of a Gaussian derivative calculation. AOS can be used for any nonlinear diffusion with a variable conductance coefficient, and it has been reported [36] that it is on the order of 10 times more efficient than a usual explicit discretization for nonlinear diffusion problems.

The result of using AOS nonlinear diffusion on the test image barbara is shown in Figure 3.4. This nonlinear diffusion is evolved with the conductivity function $g(|\nabla u|^2) = \frac{1}{|\nabla u|}$, corresponding to TV flow. Notice that the texture has been removed from barbara. For example, the checkerboard pattern on the tablecloth,

the stripes on the headscarf and some of the details on Barbara's face have disappeared. The resulting image is piecewise smooth, regions being separated from each other by large-scale contours.



Figure 3.4: Example of AOS Total Variation Flow

3.5.3 Description of Parameters

As already mentioned in Section 3.5.2, γ_1 and γ_2 are spatially varying coefficients in the energy functional 3.5.1, and depend on f_{NLD} , the initial image f evolved with the nonlinear diffusion flow of the previous subsection. The coefficient γ_1 was set to $\hat{\gamma}_1 \sqrt{|\nabla f_{NLD}|}$. Here $\hat{\gamma}_1$ is a positive real constant. The values of $\hat{\gamma}_1$ chosen later (see Table 3.1) assume that the maximum gray level in the image is 255; otherwise, the values of $\hat{\gamma}_1$ have to be adjusted accordingly. Thus, the second term of Equation 3.5.1 is only activated at cartoon edges — otherwise it is very small because of a small coefficient. The term itself is large when $|\nabla u|$ is large, and $G_\sigma * |\nabla \tilde{g}|$ is small. This favours the cartoon edge staying more in the u compo-

ment, and not spreading to the v or r component where it does not belong. Alternatively, cartoon edges could have been detected in the image using the method of [60] however the nonlinear diffusion method used fits nicely into the variational framework of the entire IES algorithm and is quite quick to compute.

The second term has coefficient γ_2 , where $\gamma_2 = \hat{\gamma}_2 \text{norm} \left(\frac{|\nabla f|}{|\nabla f_{NLD}| + \epsilon_3} \right)$. Similar to $\hat{\gamma}_1$, $\hat{\gamma}_2$ is a positive real constant. The rest of the term is normalized with a *norm* function so that the maximum value is mapped to 1. ϵ_3 is a small positive constant used to ensure that the denominator of the coefficient does not vanish. This coefficient is large where $|\nabla f|$ is large and $|\nabla f_{NLD}|$ is small. This corresponds to textured regions which have an underlying smooth cartoon component. In this instance, it is desired that the high gradient be transferred to the v component, and as little as possible to the u component, because the high gradient corresponds to texture. This is precisely what occurs (as will be seen in the experimental results in Section 3.7.6) due to the construction of the γ_2 term.

3.5.4 Euler-Lagrange Equations

Adding the Improved Edge Segregation terms in Equation 3.5.1 changes the Euler-Lagrange equations from those of Equations 2.4.5-2.4.7 for all three of the functions

u, g_1 and g_2 . The new PDEs to solve are

$$\begin{aligned}
u = & f - \partial_x g_1 - \partial_y g_2 + \frac{1}{2\lambda} \operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right) \\
& - \gamma_1 \left[\frac{d}{dx} \left(\frac{u_x}{|\nabla u| (G_\sigma * |\nabla \vec{g}|)} \right) + \frac{d}{dy} \left(\frac{u_y}{|\nabla u| (G_\sigma * |\nabla \vec{g}|)} \right) \right] \\
& + \gamma_2 \left[\frac{d}{dx} \left(\frac{(G_\sigma * |\nabla \vec{g}|) u_x}{|\nabla u|^3} \right) + \frac{d}{dy} \left(\frac{(G_\sigma * |\nabla \vec{g}|) u_y}{|\nabla u|^3} \right) \right] \tag{3.5.2}
\end{aligned}$$

$$\begin{aligned}
\mu \frac{g_1}{\sqrt{g_1^2 + g_2^2}} = & 2\lambda \left[u_x - f_x + \partial_{xx}^2 g_1 + \partial_{xy}^2 g_2 \right] \\
& + \gamma_1 \left[\frac{d}{dx} \left(\frac{|\nabla u| (G_\sigma * g_{1,x})}{(G_\sigma * |\nabla \vec{g}|)^3} \right) + \frac{d}{dy} \left(\frac{|\nabla u| (G_\sigma * g_{1,y})}{(G_\sigma * |\nabla \vec{g}|)^3} \right) \right] \\
& - \gamma_2 \left[\frac{d}{dx} \left(\frac{G_\sigma * g_{1,x}}{|\nabla u| (G_\sigma * |\nabla \vec{g}|)} \right) + \frac{d}{dy} \left(\frac{G_\sigma * g_{1,y}}{|\nabla u| (G_\sigma * |\nabla \vec{g}|)} \right) \right] \tag{3.5.3}
\end{aligned}$$

$$\begin{aligned}
\mu \frac{g_2}{\sqrt{g_1^2 + g_2^2}} = & 2\lambda \left[u_y - f_y + \partial_{xy}^2 g_1 + \partial_{yy}^2 g_2 \right] \\
& + \gamma_1 \left[\frac{d}{dx} \left(\frac{|\nabla u| (G_\sigma * g_{2,x})}{(G_\sigma * |\nabla \vec{g}|)^3} \right) + \frac{d}{dy} \left(\frac{|\nabla u| (G_\sigma * g_{2,y})}{(G_\sigma * |\nabla \vec{g}|)^3} \right) \right] \\
& - \gamma_2 \left[\frac{d}{dx} \left(\frac{G_\sigma * g_{2,x}}{|\nabla u| (G_\sigma * |\nabla \vec{g}|)} \right) + \frac{d}{dy} \left(\frac{G_\sigma * g_{2,y}}{|\nabla u| (G_\sigma * |\nabla \vec{g}|)} \right) \right] \tag{3.5.4}
\end{aligned}$$

The subtracted terms in the functional for IES in Equation 3.5.1 make it especially difficult to solve, or at least to solve in a provably convergent and stable manner. This is due to the fact that such terms make the functional non-convex, so that usual theorems for minimization do not apply.

In Appendix B, some elementary convex analysis is reviewed, after which it is demonstrated how IES can be solved using the Difference of Convex Functionals framework. Within this framework, the solution process is stable and converges to the correct minimum of the IES functional, however due to the difficulty of its implementation, it is left to future work.

3.6 Parallel Implementation

The Improved Edge Segregation code was originally written in MATLAB, but to test it in a parallel fashion on a multi-computer cluster, the code was rewritten using the Message Passing Interface (MPI) with C++ [61]. This was done as a proof-of-concept to show that IES can indeed be implemented in parallel.

Since the IES code was written in C++, existing C++ code was used from the Tornado 1.2 classes included with the SourceForge project called `restoreInpaint` [62]. These classes include implementations of many variational and other image restoration algorithms as well as many mathematical routines and image I/O functions for various image file formats. Code from the `libpng` library [63] was added, and modified, to read in PNG format files, since this was not included in `restoreInpaint`. The Additive Operator Splitting implementation of nonlinear diffusion from the Tornado classes was used for the calculation of f_{NLD} in the parallel implementation of IES.

The parallel IES code written was based on code found in [64] for the Jacobi solution of the heat equation. The image domain is partitioned using a 1D partition into horizontal sections as shown in Figure 3.5. Each partition corresponds

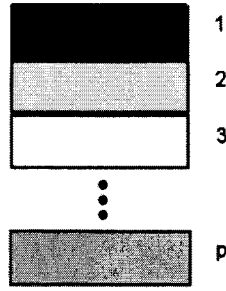


Figure 3.5: 1D Block Partitioning of Image

to one processor. Supposing that there are p processors in the parallel computer, there will also be p partitions. At each iteration it should be noted that only one row of the cartoon component u each from the top and bottom rows of each section are sent to neighbouring sections (except for the very top and bottom sections for which there is only one neighbouring section), while $\lceil \sigma \rceil$ rows each of both g_1 and g_2 from the top and bottom of every section are sent to neighbouring sections. Here σ is the radius of the Gaussian filter G_σ used to blur g_1 and g_2 in the defining functional for IES (Equation 3.5.1). This is because for any pixel in the image domain, including the inter-section boundaries, at any iteration, a window of the given radius is needed.

Blocking send and receive calls were employed for communication between computers, with the MPI C++ function `MPI_Sendrecv`. If nonblocking send and receive calls were to be used, the code would likely run faster. With nonblocking calls and suitable hardware, it is possible for example for data to be sent by one process at the same time as computations on that process' node are executed, something not possible with blocking calls. This is left for future work. For the code written for this section, the number of processors is set with a single variable, and thus the

code is scalable. Null processes are used so that there are no special cases for the processes at the top and bottom of the image. A diagram of the communication structure of the program is shown in Figure 3.6.

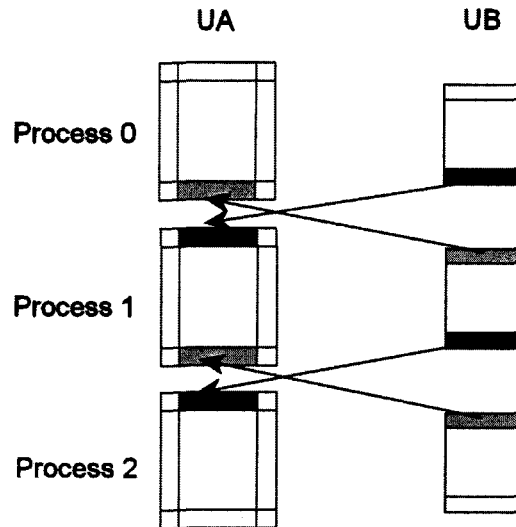


Figure 3.6: Diagram of MPI Communication Structure for IES

In Figure 3.6, there are two image data arrays per process, UA and UB. These can be assumed to correspond to the function u , which is the cartoon component of the image decomposition. Two image data arrays are necessary because we are assuming the use of Jacobi iterations, so that the original data cannot be overwritten. For the purposes of illustration in Figure 3.6, it is assumed that there is one row of overlap between the image data UA of one process and that of the process immediately below, if such a process exists. Image decomposition for one process from one iteration operates on the corresponding UA array as input, for example (there are actually three input image data arrays operated on per process, corresponding to the three functions u , g_1 and g_2 of the decomposition), and the output

is written to the corresponding UB array. The UB output is the calculated u for the next iteration. Then this UB output is copied to the interior of the UA data array of the same process to be used as input for the next iteration. Since the calculation of the decomposition for the next iteration at the boundary of the interior of the UA data array depends on data just calculated from the neighbouring processes, the data on the boundaries of the adjacent UB processes is also required. Therefore, this data is sent to the boundary of the UA data array from the boundary of the UB data arrays of these neighbouring processes. The transmission of this data is represented in Figure 3.6 by the arrows going from the boundaries of the UB data arrays to the boundaries of the neighboring UA data arrays.

3.7 Results

3.7.1 Test Set

The test images used in this chapter's experiments are shown in Figure 3.7. The first is a slightly cropped barbara image (256x256 pixels), the second is a zoomed portion of the entire barbara image (171x147 pixels) which is called barbzoo, while the third test image is mosaic (160x160 pixels), a mosaic of two Brodatz textures [65, 66] (brick on top and scale on the bottom). The image barbara was chosen because it is very "busy", with a lot of small-scale texture throughout the image, along with some cartoon parts, e.g. the table leg. The image barbzoo was selected since it had some medium-scale texture (the top of Barbara's headscarf). The image mosaic was chosen because it could be used for texture segmentation



Figure 3.7: Images used to test image decomposition: (a) barbara, (b) barbzoo, (c) mosaic, (d) woodangle and (e) cactus

once the discrimination was complete. The fourth test image, woodangle (256x256 pixels), was chosen for the same reason as well as due to the fact that the texture (a Brodatz texture) is the same in both parts of the image, with one a 90 degree rotation of the other. Also, the texture is less structured and more random and sparse than in the other images in the test set. Finally, cactus (225x300 pixels) was selected because it was larger than the other images and was simple, other than

the cactus plant itself. As can be seen, it is a picture of a cactus in a pot. All the images were in Portable Network Graphics (PNG) format, except for *barbara*, which was in ASCII Portable Gray Map (PGM) format. They were also all 8-bit grayscale images.

3.7.2 Jacobi vs. Gauss-Seidel Iterations

In the experiments conducted for this chapter, and in this thesis as a whole, Jacobi iterations were looked upon more favourably than Gauss-Seidel iterations for the discretization of the solution of any systems of PDEs. This is because Jacobi solvers are more easily parallelizable than Gauss-Seidel ones. In fact, since the values of all of the grid points of the previous iteration have already been computed, and only these points are being used, one processor can be used separately for each grid point. In a true parallel implementation, this would lead to a substantial time savings.

On the other hand, it is impossible to parallelize Gauss-Seidel iterations to this extent, because previously computed grid points from the current iteration also have to be used, so that there is some dependence between various grid points of the current iteration. The best parallelization for Gauss-Seidel uses a red-black colouring scheme, and leads to a speedup ratio of $\frac{N}{2}$, where N is the total number of points in the 2-D grid. An illustration of the red-black colouring scheme is shown in Figure 3.8, taken from [67]. This colouring is based on a Gauss-Seidel solution of the heat equation (Equation 2.2.6). The values of all the nodes which are of the same colour can be calculated in parallel because there is no dependency

within each group of nodes. First, all the red node values are computed, and then all the black node values using the just-calculated red node values. This alternation continues until convergence. So this ends up being at least twice as slow as a parallel Jacobi implementation. In practice, it is even worse for the case of decomposition, due to discretization of the mixed derivatives of g_1 and g_2 . If these are discretized in such a way that for a given grid point, not only the grid points immediately vertically or horizontally adjacent are used in the computation of its value, but others as well, then a red-black colouring as shown in the figure will not be possible. For the discretizations in the paper by Vese and Osher [2], this is always the case.

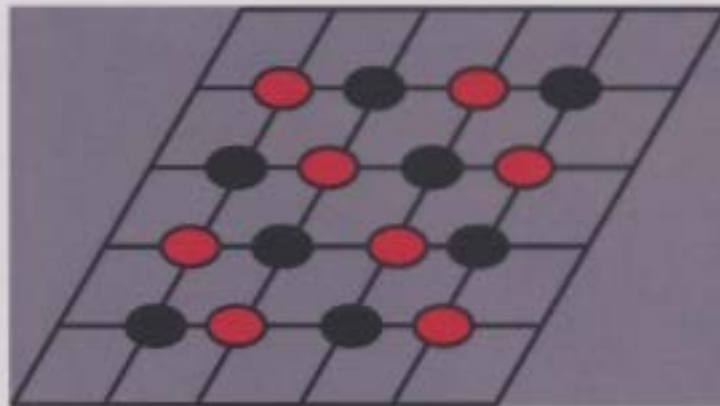


Figure 3.8: Illustration of Red-Black Ordering for Gauss-Seidel Iterations, taken from [67]

It was found by trial and error that for Gauss-Seidel discretization of V-O decomposition, at least 5 colours would be necessary. A diagram of the colouring is shown in Figure 3.9.

For each of the functions u , g_1 and g_2 , a 3x3 grid, or stencil, as it is commonly called, is shown. This stencil shows which grid function values are used in the finite difference computation of the iterative solution of the PDE at the central

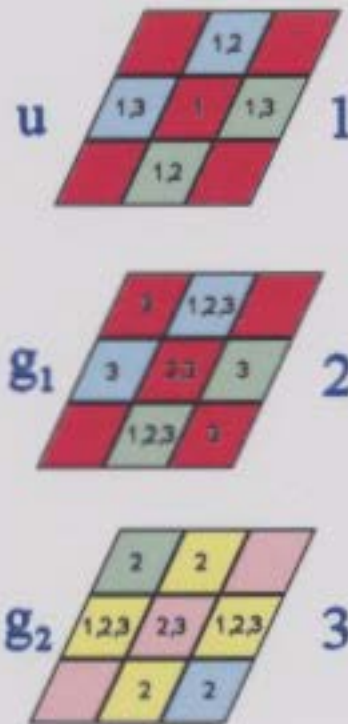


Figure 3.9: 5-Colouring of Nodes for V-O Decomposition

square in the grid. The number 1 on a square refers to the function u , the number 2 to g_1 and the number 3 to g_2 . Additionally, the top stencil, labelled 1, corresponds to the computation of the function u , the middle stencil, labelled 2, to the computation of g_1 and the bottom stencil, labelled 3, for g_2 . A number on a square in a stencil means that the value of the function corresponding to the number on that square is used in the computation of the function value for the central square of the function that the grid corresponds to. It should be noted that each stencil is tiled so that it spans over the whole domain of the image.

Though a proof has not been found that there is no colouring of the grids for Gauss-Seidel computation of the V-O model solution which requires fewer than 5 colours, no counterexample which uses four colours or less has been found after

an extensive search, and it is strongly suspected that 5 is indeed the minimum. The criterion that was used to show that the colouring indeed leads to each processor only computing grid values which are independent of each other was that no grid values for any function coloured with a given hue should appear both on the left-hand side of a computation and a right-hand side. If a square in a grid for a given function, say h , to make the argument generic, is of a certain colour, that means that grid value for h is used on the left hand side of a computation. For each colour used up to that point, a separate 3x3 colour grid is used to ensure that function values for that colour don't appear on both left and right hand sides of iterative equations. Then remembering that both the 3x3 colour grids and stencils are periodically tiled to cover the whole image, when a square in a function stencil is coloured with a given hue, the 3x3 stencil for that function is superimposed (periodically repeated horizontally and vertically) on that position in the colour grid, and the numbers in the stencil placed on the matching squares in the colour grid. This is repeatedly done, while keeping in mind the rule that if there is a number at a certain square in a colour grid, then that hue cannot be used to colour that position in the stencil for the function corresponding to that number.

But even if at least 5 processors are necessary for optimal speed-up of Vese-Osher decomposition, there is another problem inherent with Gauss-Seidel iteration for the purposes of solving the equations for this model, or for Gauss-Seidel iterations in general. The communication costs between processors will be prohibitively high due to the fact that for any block or section of the image, there will have to be "vertical" communication between the same pixels being processed by different processors for different functions, i.e. u , g_1 and g_2 . This is in addition to the com-

munication between sections, as shown in Figure 3.6.

3.7.3 Experimental Approach

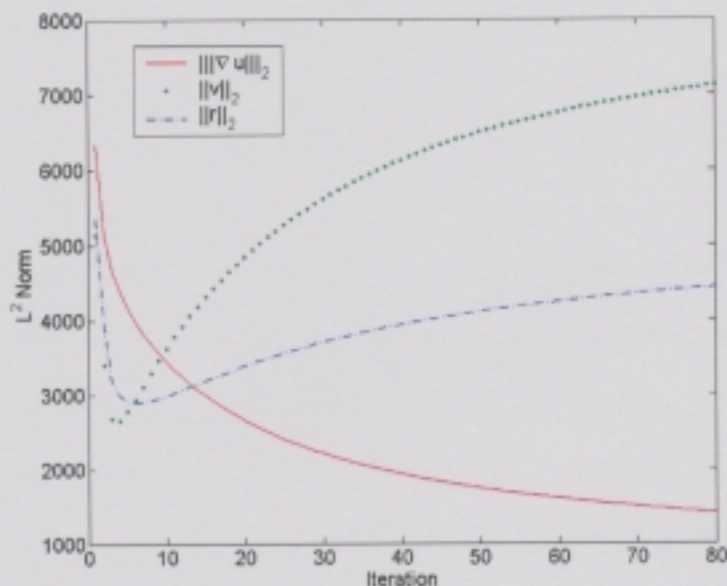


Figure 3.10: Progression of Total Variation of u and L^2 -Norms of v and r vs. Iteration Number for V-O Model on mosaic Test Image

All the different decomposition algorithms (V-O, IES and OSV models) were run on the images of the test set.

One item of implementation concern was the stopping criteria for these iterative methods. In Osher, Solé and Vese [4], the authors ran their algorithm until the L^2 -norm of v became greater than that of the TV flow, so we follow a similar procedure. A similar stopping condition could not be used, especially for the IES model, because by construction, this method does not place cartoon edges in the v component, and thus the L^2 -norm of v will be less than that of Vese and Osher for most images. This is despite the fact that more texture is placed in the v component

away from the cartoon edges.

The approach that is taken is that first the decomposition of Vese and Osher [2] is run on the image $u(x, y)$. Suppose that, starting from the initial image, $u_0(x, y) = f(x, y)$, the evolved image is $u_i(x, y)$ at the i th iteration. Then V-O decomposition is continued until the following condition is reached:

$$|TV(u_n) - TV(u_{n-1})| = TV(u_{n-1}) - TV(u_n) < \Delta TV(u),$$

where $TV(u) = \int_{\Omega} |\nabla u| dx dy$, and $\Delta TV(u)$ is a suitably selected constant. The first equality holds because the process was found experimentally to be TVD (Total-Variation Diminishing), see e.g. Figure 3.10. Then, IES is also run until this condition is met. Say that for IES, n_{IES} iterations are required until this occurs. The algorithm of Osher, Solé and Vese [4] is run until the first iteration k where $\|v_k^{OSV}\|_{L^2} < \rho_v \cdot \|v_{n_{IES}}\|_{L^2}$ with ρ_v a previously determined constant greater than or equal to 1, which is set to 1.1 for all test images. This gives a fixed procedure which conforms well to qualitative human perception of when the iterative decomposition has essentially converged. However it is possible that better stopping criteria exist.

In the OSV decomposition scheme, a gradient projection method was used, so that the fidelity parameter λ varied with each iteration. The parameter λ was found by assuming that at equilibrium, $\|v_{OSV}\|_{L^2} = \|v_{IES}\|_{L^2}$. Using integration by parts, it was determined that

$$\lambda = -\frac{1}{2K} \int_{\Omega} (f_x - u_x) \frac{\partial}{\partial x} \operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right) + (f_y - u_y) \frac{\partial}{\partial y} \operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right) dx dy, \quad (3.7.1)$$

with $K = \|v_n^{IES}\|_{L^2}$. To see why the dynamic formula for λ in Equation 3.7.1 holds

true, recall that the defining partial differential equation for the evolution of u is

$$u_t = -\frac{1}{2\lambda} \Delta \operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right) - (u - f). \quad (3.7.2)$$

Thus, at equilibrium, the right-hand side of the equation will equal zero, meaning that

$$f - u = \frac{1}{2\lambda} \Delta \operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right). \quad (3.7.3)$$

If each side of the above equation is multiplied by $f - u$, then,

$$(f - u)^2 = (f - u) \frac{1}{2\lambda} \Delta \operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right), \quad (3.7.4)$$

and subsequently integrating each side of the above equation over the entire image domain Ω , and using the fact that $\|v\|_{L^2}^2 = \int_{\Omega} (f - u)^2 dx dy$, the following equation is obtained

$$\|v_n^{IES}\|_{L^2}^2 = \frac{1}{2\lambda} \int_{\Omega} (f - u) \Delta \operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right) dx dy. \quad (3.7.5)$$

Using integration by parts, the integral above can be replaced by

$$-\frac{1}{2\lambda} \int_{\Omega} (f_x - u_x) \frac{\partial}{\partial x} \operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right) + (f_y - u_y) \frac{\partial}{\partial y} \operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right) dx dy \quad (3.7.6)$$

This is found using standard integration by parts, and the fact that $\frac{\partial}{\partial x} \operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right)$ and $\frac{\partial}{\partial y} \operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right)$ are both equal to zero in the OSV model [4]. Then Equation 3.7.1 follows upon rearrangement of Equation 3.7.6.

After each test image was decomposed, discrimination was attempted between the textures in the image with Active Contours without Edges (ACWE) on either

the absolute value of the g_1 component or g_2 component. ACWE texture discrimination was tested for only two images, mosaic and woodangle. This is because these are the only images in the test set which each consist of two spatially separated textures. If there were more than two textures, then more than one level set function would be required. Although this is possible, it would make the code more complex.

3.7.4 Choice of Parameters

In this section, the parameters selected for the various image decomposition methods are listed. It may be possible in the future to use a heuristic similar to that found in [59], to automatically select the parameters needed for a given image so that it is not necessary to experimentally determine these parameters values. All the values of λ and μ were in the same general ranges as those found in the paper by Vese and Osher [2].

For all the images, the time step for AOS nonlinear diffusion was $\Delta t = 6$, and the regularization constant ϵ was set to 1. The standard deviation σ_{NLD} used for the calculation of Gaussian derivatives was $\sigma_{NLD} = 2$ for all images. The number of iterations that AOS diffusion was run on each image is shown in Table 3.1, along with the other parameter choices.

For IES on all the test images, the regularization parameters, ϵ_i ($1 \leq i \leq 3$), were set as follows: $\epsilon_1 = 0.5$, $\epsilon_2 = 1$ and $\epsilon_3 = 1$.

There are only two main parameters for ACWE that need to be specified, the grouping parameter μ and Δt , the time step. To avoid confusion with the param-

Image name	μ	λ	$\hat{\gamma}_1$	$\hat{\gamma}_2$	σ	$\Delta TV(u)$	n_{TV}
barbara	0.05	0.05	0.12	1.2	1	1000	15
barbzoom	0.05	0.02	0.1	0.5	2	700	20
mosaic	0.05	0.02	0.12	0.6	2	700	35
woodangle	0.05	0.03	0.1	0.6	2	700	15
cactus	0.05	0.05	0.12	0.6	2	800	15

Table 3.1: Parameter choices for IES on test images

Image name	Decomp. Alg.	μ_{ACWE} (x255 ²)	Δt_{ACWE}	$niter_{ACWE}$	t_{ACWE}
mosaic	V-O	0.2	0.1	220	58.03
mosaic	IES	0.4	0.1	171	45.11
woodangle	V-O	-	-	-	-
woodangle	IES	0.5	0.1	940	636.20

Table 3.2: Decomposition Algorithm, parameter choices, number of iterations required, and timing results for ACWE Texture Discrimination

Image name	λ_{analyt}^0	n_{analyt}^0	$\ v\ _{2,IES}$	Δt_{analyt}
barbara	0.02	10	1995	0.002
barbzoom	0.02	10	2937	0.0008
mosaic	0.02	10	5305	0.0005
woodangle	0.01	3	3848	0.00003
cactus	0.02	10	2111	0.005

Table 3.3: Parameter choices for OSV decomposition

eters for the decomposition phase, the parameter names μ_{ACWE} and Δt_{ACWE} are used. For this chapter’s tests, a stopping condition for convergence of the active contours algorithm needed to be specified. Discrimination was run for both IES and V-O $|g_1|$ subcomponents until all “specks” were gone, and there were only two large contiguous regions remaining. The parameter choices and resulting timings for ACWE texture discrimination are shown in Table 3.2. Finally, in Table 3.3 are shown the various parameter choices for OSV decomposition as well as the L^2 -norms of the texture components v experimentally obtained via IES on the test images. Recall that these L^2 -norms were required for the dynamic selection of λ in the OSV test runs.

3.7.5 Decomposition Quality Measures

It is possible to visually compare the quality of two series of decomposition components obtained from different algorithms or different parameter sets with each other. A decomposition result is of higher quality when cartoon information, as found in cartoon edges, is kept in the cartoon component u , and does not spread to the texture component v . Similarly, texture information should be kept in the texture component v , and not be present in the u component. A visual comparison, keeping these two quality guidelines in mind, is subject to interpretation, especially when results are similar, and it becomes difficult to quantify how much better one decomposition result is over another. Thus it would be beneficial to develop one or more quantitative decomposition quality measures. A series of such simple decomposition quality measures is now given and described.

In order to determine how good a given decomposition result is, it is important to first determine which edges in the original image are cartoon, or large-scale edges, and which are texture, or smaller-scale edges. One way in which this can be done by using the total variation (TV) flow of Rudin, Osher and Fatemi [38], derived from the total variation energy given earlier in Equation 2.4.1. The TV flow can be implemented efficiently with an Additive Operator Splitting scheme, which is known to be unconditionally stable [36]. The output of the TV flow for the decomposition quality measure in this paper, f_{TV} is determined by running Additive Operator Splitting for 4 iterations with a timestep Δt_{AOS} equal to 2. This total variation flow eliminates small-scale texture and maintains large-scale cartoon edges, though the latter may be blurred. Thus, texture edges can be determined as being located at pixels with a high gradient magnitude in the original image f , but a low gradient magnitude in the image f_{TV} output from the TV flow, relative to the original high gradient magnitude in f .

Cartoon edges on the other hand, are determined as being located at pixels with high gradient magnitude both in the original image and the image evolved with the TV flow. By setting a gradient magnitude threshold gmt and a gradient ratio parameter r_{gmt} , the set of cartoon edge pixels C in the image f is defined to be

$$C = \{(x, y) \text{ such that } |\nabla f(x, y)| > gmt \text{ and } |\nabla f_{TV}(x, y)| > r_{gmt} |\nabla f(x, y)|\}$$

and the set of texture edge pixels T in f to be

$$T = \{(x, y) \text{ such that } |\nabla f(x, y)| > gmt \text{ and } |\nabla f_{TV}(x, y)| \leq r_{gmt} |\nabla f(x, y)|\}.$$

The value of the threshold gmt used in these two definitions is selected to be 20, assuming a maximum image intensity of 255, and the gradient ratio parameter $r_{gmt} = 0.6$.

Then the cartoon quality measure CQ of a decomposition result set (u, v) is calculated as being the average value of $\frac{|\nabla u|}{|\nabla f|}$ in C , and the texture quality measure TQ of this set is defined to be the average value of $\frac{|\nabla v|}{|\nabla f|}$ in T . In the calculation of TQ , one-sided forward differences are used in the computation of the gradient magnitudes of v and f , since texture edges are very small scale, and a central difference discretization may not adequately detect these edges. Clearly, in general if CQ and TQ are higher in value, then cartoon and texture edges are better preserved in their respective components, and the decomposition is of superior quality.

One potential problem with the cartoon measure CQ occurs when there is texture on one or both sides of a cartoon edge. In this case, if this texture is better extracted to the v component of the decomposition, which indicates better decomposition quality, then it will be more blurred on that/those side(s) of the edge in the u component, and potentially the cartoon edge will actually be weaker. This leads to a lower CQ value. Thus, a new cartoon quality measure CQ' can be defined as the average value of $\frac{|\nabla u|}{|\nabla f|}$ on the set of cartoon edge pixels in C which are at a distance of over 2 pixels away from any pixels in T , the set of texture edges. This avoids the problem just described, though the number of cartoon edge pixels in the set over which the average in the cartoon quality measure is computed, is reduced. Denote this set of cartoon edge pixels by C' . An example of the two sets of detected cartoon edges C and C' , as well as the set of detected texture edges T , is shown in white in Figure 3.11, corresponding to calculation of these sets for the test image barbara.

The decomposition quality measures CQ , CQ' and TQ are now used in the next subsection to compare results obtained from V-O and IES decomposition on the images in the test set.

As in [68] by Shahidi and Moloney, it is desired to reduce the presence of cartoon edges in the texture component of the outputs from image decomposition schemes.

3.7.6 Serial Decomposition Results and Discussion

As can be seen in Figure 3.12, the V-O image decomposition model leaves many cartoon edges in the texture component v . Cartoon edge strength is considerably lower in the v component from the proposed IES decomposition in Figure 3.13.

Image name	$niter_{VO}$	$t_{VO}(s)$	$niter_{IES}$	$t_{IES}(s)$	$niter_{analyt}$	$t_{analyt}(s)$
barbara	50	55.00	43	48.80 (+4.75)	314	66.06
barbzoom	40	20.09	52	24.55 (+0.95)	948	67.06
mosaic	101	45.92	99	41.00 (+1.80)	1447	95.98
woodangle	63	68.48	78	88.92 (+4.77)	31417	6963.2
cactus	33	39.80	44	49.27 (+3.05)	100	19.84

Table 3.4: Number of Iterations and CPU Time required for Vese-Osher vs. proposed algorithms on Test Set

The v component of this IES decomposition actually has more texture in some parts (for example the top of the tablecloth) than Vese and Osher’s. On the other hand, there is an obvious reduction of cartoon edges in the texture component v (e.g. the legs of the table or Barbara’s arm). The reduction of cartoon edges in the texture component is very desirable because it indicates better decomposition quality — this cartoon information should only appear in the u component, as has been stated in several papers, e.g. [4] and [40]. As a result, it is seen that the edges in the u component of barbara in Figure 3.13 (IES) are sharper than the correspond-



(a) Detected Cartoon Edges (C)



(b) Detected Cartoon Edges (C')



(c) Detected Texture Edges (T)

Figure 3.11: Detected Cartoon and Texture Edges in Test Image barbara Used for Calculation of Decomposition Quality



(a) u component



(b) v component



(c) r component

Figure 3.12: V-O Decomposition [2] ($\mu = \lambda = 0.05$) of barbara after 50 iterations



(a) u component



(b) v component



(c) r component

Figure 3.13: IES Decomposition of barbara ($\mu = \lambda = 0.05, \hat{\gamma}_1 = 0.12, \hat{\gamma}_2 = 1.2, \sigma = 1$) after 43 iterations

ing edges in Figure 3.12 (V-O). There is slightly more information remaining in the r component of the proposed IES scheme; it is possible that the energy in the r component could be reduced with suitable parameter selection, e.g. a higher μ , but this has not been done for this thesis. Overall, a better decomposition is achieved by IES over Vese-Osher.

Not only is there a better quality result, but it is achieved in approximately the same amount of time as the standard V-O decomposition. This can be seen in Table 3.4 where CPU times on a 2.6 GHz Pentium 4 desktop with 1 GB of RAM are reported. For IES, the time required for the PDE solution proper is given in seconds followed in brackets by the time needed for the pre-computation of f_{NLD} using AOS nonlinear diffusion flow. Recall that these times are despite the increased terms in the energy functional of IES over Vese-Osher.

Similar statements can be made for the image mosaic. The brick edges on the top of the mosaic are considered to be cartoon edges since they are very large scale. For the V-O decomposition (Figure 3.14), these edges are quite evident in the v component. However, the IES result (Figure 3.15) is much improved. The bottom part of the u component of the mosaic for IES is very similar to Vese and Osher's with most of the scale texture smoothed or removed. The top section of the u component of the mosaic is sharper in IES than in V-O, since the cartoon edges are retained more in the correct component. For instance, the brick edges in the v component are much more prominent with V-O decomposition as opposed to IES. Once again, fewer iterations and comparable CPU time are required for a superior result (see Table 3.4).

The auxiliary functions g_1 and g_2 for V-O decomposition are shown in Figure

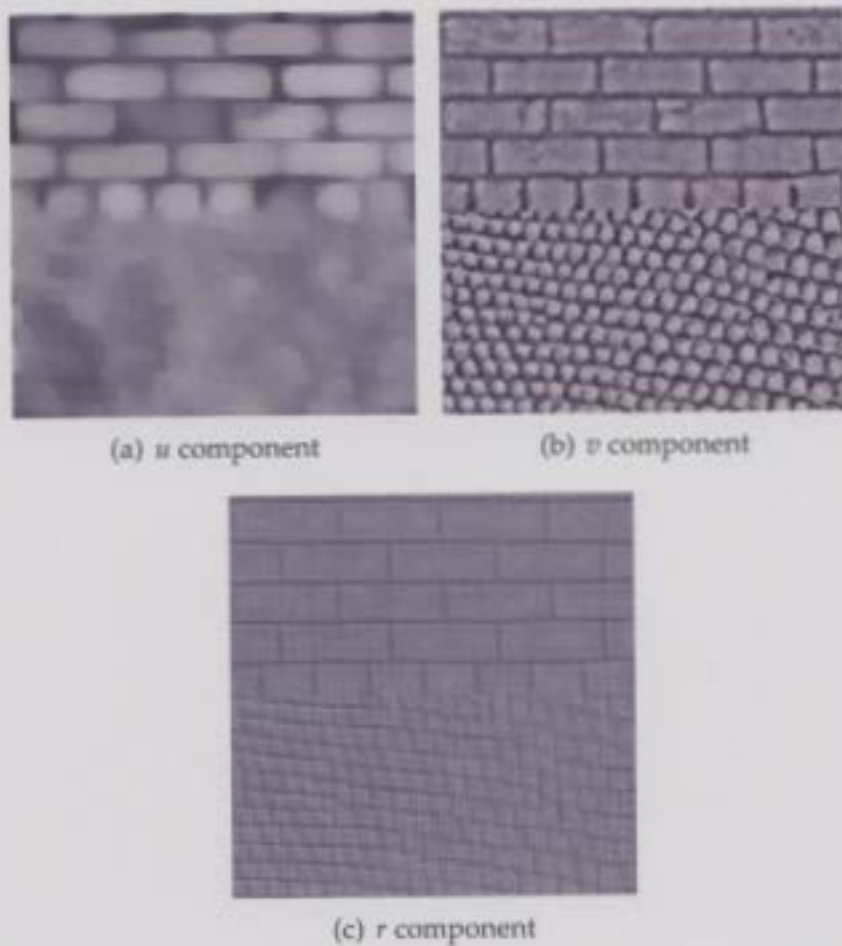


Figure 3.14: V-O Decomposition [2] (u , v , and r) of mosaic ($\mu = 0.05, \lambda = 0.02$) after 101 iterations

3.16, where it can be seen that as according to the analytical result in Appendix A, g_1 largely contains the horizontal energy of the texture, while g_2 mostly contains the vertical energy. Note that for visualization purposes, 130 has been added to g_1 and g_2 , where the maximum possible image intensity corresponds to a value of 255.

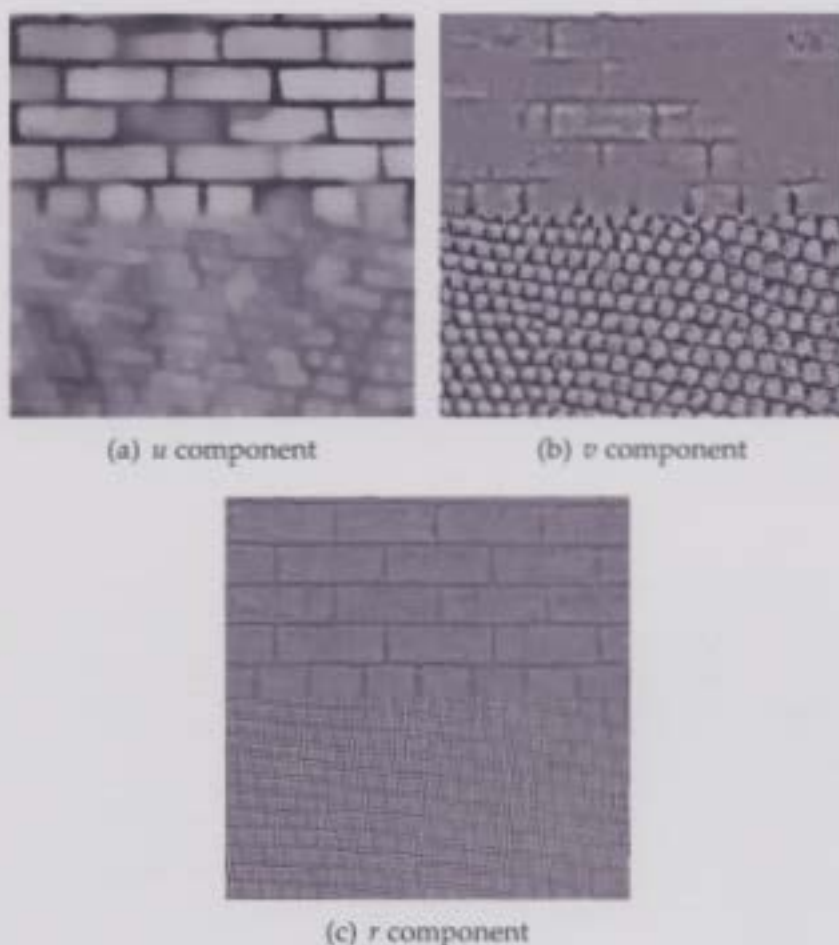


Figure 3.15: IES Decomposition of *mosaic* ($\mu = 0.05, \lambda = 0.02, \hat{\gamma}_1 = 0.11, \hat{\gamma}_2 = 0.75, \sigma = 2$) after 99 iterations

Finally for the cactus image, the improvement of both IES and the OSV result over V-O decomposition is seen once again in Figures 3.17, 3.18 and 3.19. Especially for the IES result, it is observed that hardly any of the edges of the pot hold-

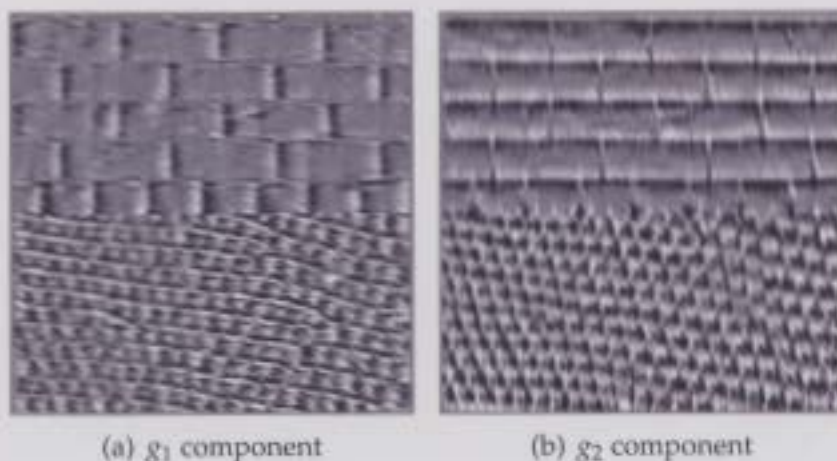


Figure 3.16: V-O Decomposition [2] (g_1 and g_2) of mosaic ($\mu = 0.05, \lambda = 0.02$) after 101 iterations

ing the cactus show up in the u component.

The OSV results are better in quality, but usually took more time than either V-O or IES. This is due to the fact that often a very small stepsize must be chosen for the explicit time stepping to ensure stability.

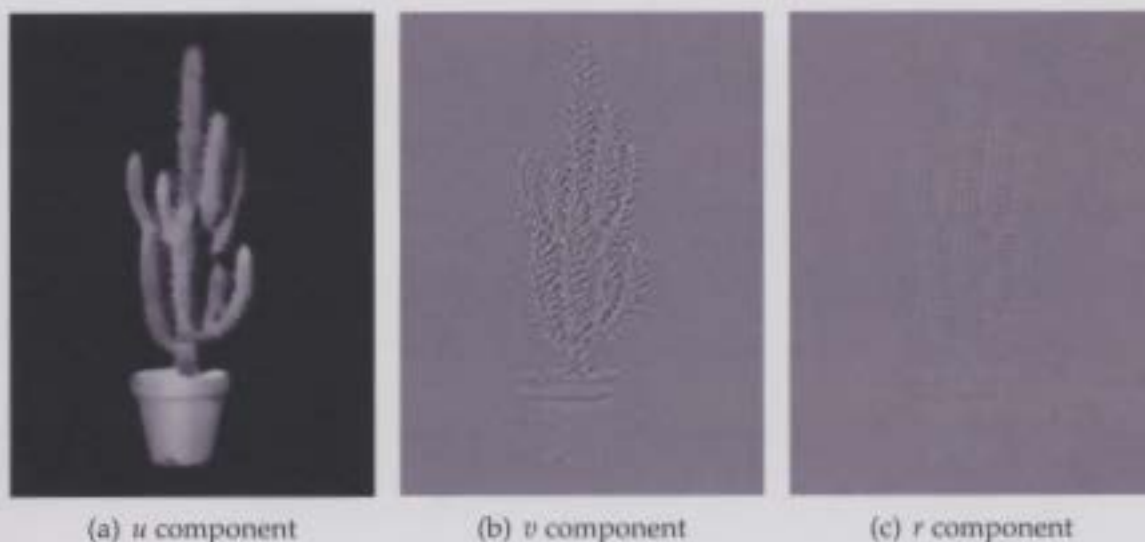


Figure 3.17: V-O Decomposition [2] of cactus ($\mu = 0.05, \lambda = 0.05$) after 35 iterations

Next, quantitative results comparing the Vese-Osher and IES decomposition al-

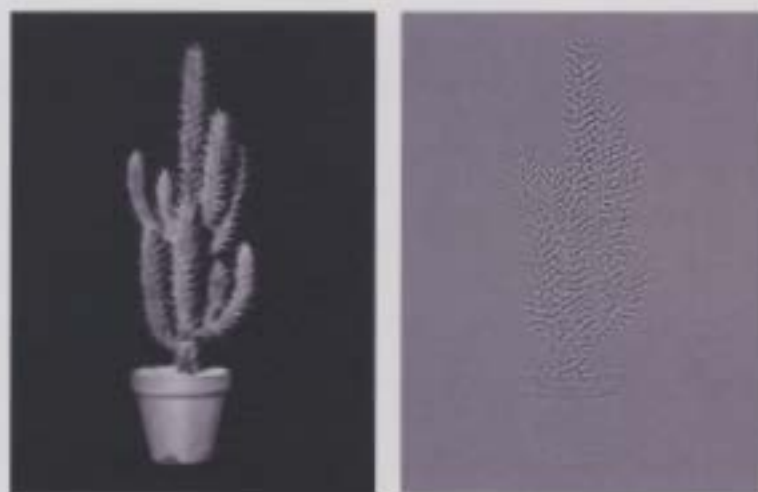


(a) u component

(b) v component

(c) r component

Figure 3.18: IES Decomposition of cactus ($\mu = 0.05, \lambda = 0.05, \hat{\gamma}_1 = 0.12, \hat{\gamma}_2 = 0.6, \sigma = 2$) after 44 iterations



(a) u component

(b) v component

Figure 3.19: OSV Decomposition of cactus after 100 iterations

gorithms using the decomposition quality measures in Section 3.7.5 are given.

3.7.6.1 Decomposition Quality Measure Results

Test Image	Decomp. Alg.	CQ	CQ'	TQ
barbara	V-O	0.5727	0.6501	0.7974
	IES	0.8670	0.9327	0.6000
barbzoom	V-O	0.1416	0.1876	0.8230
	IES	0.2274	0.3013	0.8050
mosaic	V-O	0.2596	0.4379	0.8020
	IES	0.4346	0.7635	0.7793
wood	V-O	0.1605	0.3980	0.6435
	IES	0.3049	0.5826	0.7578
cactus	V-O	0.5060	0.7774	0.8152
	IES	0.7558	0.9684	0.5897

Table 3.5: Decomposition Quality Measures of Test Images for V-O and IES Decomposition Models

Table 3.5 lists the calculated cartoon and texture quality measure values (defined in Section 3.7.5) for the cartoon and texture components of the Vese-Osher and IES decompositions of the test images used in the experiments in this chapter. Observe that the cartoon quality measures CQ and CQ' are much higher for the IES decompositions than the V-O decompositions. On the other hand, the texture quality measure TQ is significantly lower for the IES decompositions of barbara and cactus than the corresponding V-O decompositions. However, for all the test images, including barbara and cactus, the average value of the cartoon quality measure CQ and the texture quality measure TQ is higher for the IES decomposition results than the corresponding V-O decompositions. For barbara and cactus, this deterioration in texture quality was found to be due to the fine scale nature of the texture in those images. Observe that for the test image wood, there was actually a

substantial improvement in the texture quality measure TQ as well as the cartoon quality measures CQ and CQ' .

To improve the texture quality measure values, preliminary promising experiments have been performed, replacing the second additional term

$$- \gamma_2 \int_{\Omega} \frac{G_{\sigma} * |\nabla \vec{g}|}{|\nabla u| + \epsilon_2} dx dy \quad (3.7.7)$$

in Equation 3.5.1 with the following term

$$- \gamma_2 \int_{\Omega} \frac{v^2}{|\nabla u| + \epsilon_2} dx dy = - \gamma_2 \int_{\Omega} \frac{(g_{1,x} + g_{2,y})^2}{|\nabla u| + \epsilon_2} dx dy, \quad (3.7.8)$$

which also serves to emphasize texture in the texture component, but which also has been found to work well for fine-scale texture, unlike the term in Equation 3.7.7. Further experiments with this revised term are left to future work.

3.7.7 Parallel Decomposition Results and Discussion

The parallel MPI code that was written for Improved Edge Segregation and that was briefly discussed in Section 3.6, was run on a 4-node Linux cluster with version 5 of the Clustermatic package installed. Clustermatic is based on LinuxBIOS (which replaces the usual BIOS bootstrap procedure) and BProc, short for the Beowulf Distributed Process Space, which allows processes to be started on any of the machines in the cluster easily [69]. Each node of the cluster was a PC with 2 gigabytes of memory and a 3.0C GHz Pentium IV processor. The nodes were connected together with a 1 gigabit per second Dell router. The times required for

running IES with the test images as input on the cluster are shown below in Table 3.6. The same number of iterations of IES were run as for the serial case for each of the test images, these numbers of iterations given in Table 3.4.

Image name	Time on 1 node of cluster (s)	Time Required on all 4 nodes (s)
barbara	5.89	1.52
barbzoom	4.07	1.25
mosaic	8.20	2.46
woodangle	16.42	4.34
cactus	8.95	2.57

Table 3.6: CPU Time Required for IES Decomposition on Parallel Beowulf Cluster

Both the running times on one node of the cluster (as measured with the UNIX clock command), and on all of the nodes of the cluster (using logfiles as soon explained) are given in Table 3.6. There is some time required for initialization, e.g. for reading in the input image to be decomposed, and performing nonlinear diffusion to determine cartoon and texture edges, but after this, as expected, since there are 4 nodes in the cluster, the time required for the actual decomposition on all 4 nodes is roughly a quarter of that required for the actual IES decomposition on just one node. Also, much less time was required even when the code was run on just one node as compared to the running times reported in Table 3.4. This is because the running times in Table 3.4 were found on a computer with a slower processor (2.6 GHz vs. 3.0C GHz) and less memory, and also importantly, were measured in a MATLAB environment, which is interpreted and not compiled, and is thus substantially slower.

On the cluster, log files were created using the `mpilog` command-line option to `mpirun`, and were viewed graphically using Jumpshot-4 [70], which comes with the MPICH package [61] for running MPI programs on the cluster. From the log files, it

was seen that a negligible amount of time was required for the passing of messages between cluster nodes, and the bulk of the time was used either for preprocessing, the actual execution of the IES parallel decomposition or waiting for messages to be received from other nodes because of unevenness in the running times between the nodes.

An example snapshot of the Jumpshot viewer with the log file for the image barbara is shown in Figure 3.20. The green block to the left of the graph shows pre-computation time for MPI to be initialized, etc. The numbers 0, 1, 2 and 3 can be seen in the left pane of the captured window. The white vertical arrows which appear periodically in the figure correspond to messages being passed between the cluster nodes with numbers in the left pane at the same horizontal position as that arrow's endpoints. As can be seen from the Jumpshot viewer snapshot, the actual passing of messages is extremely rapid, and most of the overall time is taken with the actual calculation of the IES decomposition of barbara; this being the case with the other images as well.

3.7.8 Texture Discrimination

3.7.8.1 Background

As discussed in Chapter 1, the characteristic that distinguishes texture discrimination from segmentation is that for discrimination, the textures have already been separated out spatially, with very little mixing between them. For the general segmentation problem, each texture can be spread across the entire image, making it a more difficult problem than discrimination alone.

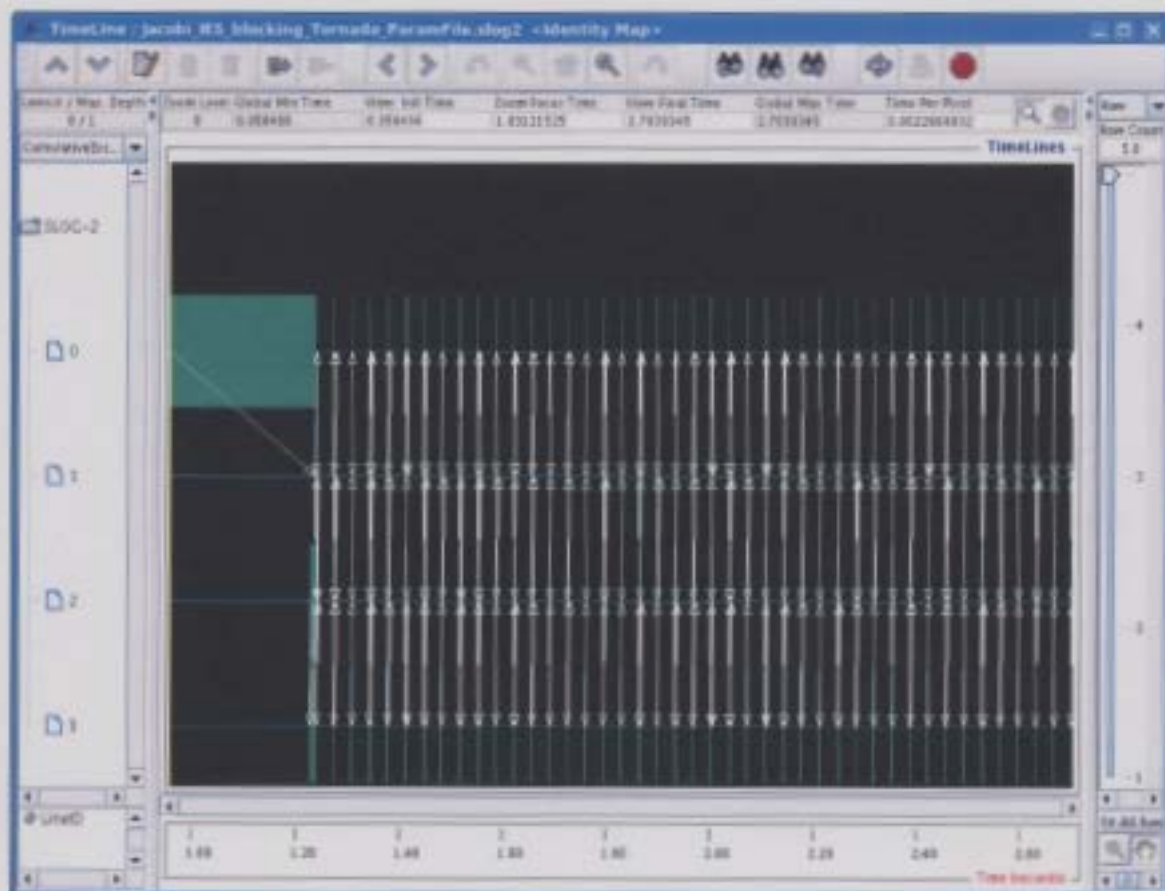


Figure 3.20: Snapshot of Jumpshot Window running with Logfile from barbara on cluster

Only the problem of texture discrimination lies in the scope of this thesis, but the approach taken can be generalized to segmentation as well, as stated by Vese and Osher [2]. As in Vese and Osher's paper [2], the texture components g_1 and g_2 are used for texture discrimination. The traditional approach to texture discrimination has been the use of Gabor filters [71]. This is, however, very computationally intensive, since the response to an entire filter bank of Gabor filters of differing orientations, scales and frequencies has to be used, forming a large number of channels. This fact is reported in [2]. However, with the use of the components g_1 and g_2 of the texture decomposition, only one or two channels need be used,

leading to great time savings.

3.7.8.2 Results

Table 3.2 gives the parameters used for the discrimination runs on the two test images *mosaic* and *woodangle*, together with the obtained timing results. For both images, the absolute value of the g_1 subcomponent was used for the discrimination.

For the image *woodangle*, it was found that the difference between the texture patches in g_1 and g_2 was so slight with V-O decomposition that texture discrimination was not possible for a wide range of parameters in ACWE . This problem was not seen with IES decomposition, as the two texture patches were more distinguishable, though discrimination did take a relatively large amount of time (app. 10 minutes). However, the discrimination could be stopped earlier without significant deterioration in discrimination quality. A rough discrimination result from an early stopping of IES may be considered to be better than the lack of any result whatsoever for V-O decomposition. The IES discrimination result is shown in Figure 3.21.

For *mosaic*, a more accurate contour between the two textures was seen when IES decomposition was used as the first step of the two-step scheme as opposed to V-O decomposition (See Figure 3.22). Additionally, this better result was accomplished in less time (45.11 seconds) than required for the discrimination from absolute value of the g_1 subcomponent V-O decomposition of the *mosaic* image (58.03 seconds).

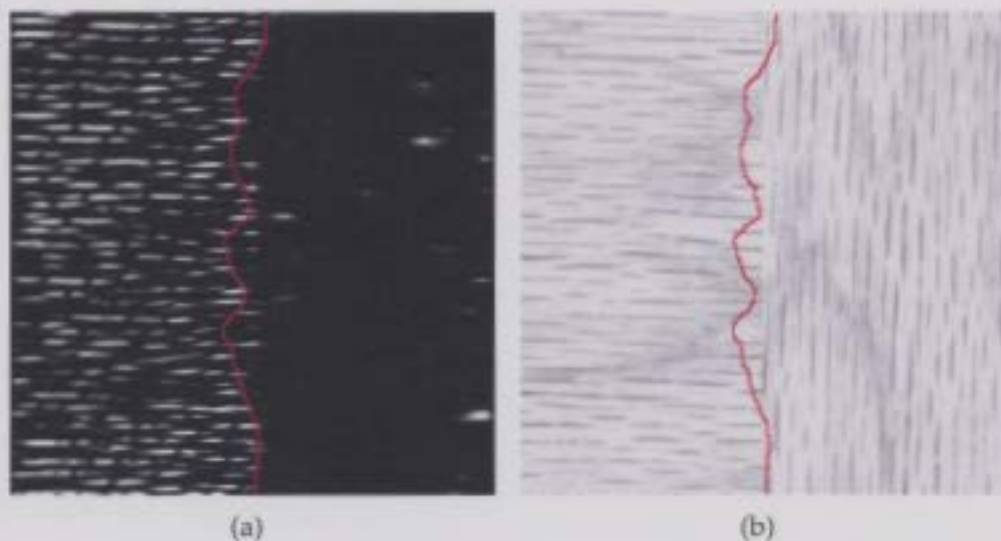


Figure 3.21: (a) ACWE Segmentation result for *woodangle* overlaid on $|g_1|$ obtained via IES decomposition, and (b) Segmentation result overlaid on original *woodangle* image for IES decomp.

3.8 Conclusions

In this chapter, three different decomposition models were presented extending that of Vese and Osher, namely the Geometric Constraint (GC) Model, the Edge Segregation (ES) Model and the Improved Edge Segregation (IES) Model. The IES model gave particularly good results, both in terms of execution time and quality of decomposition. IES also gave superior discrimination results, as compared to the V-O model, when a texture subcomponent from decomposition was fed into an Active Contours without Edges discrimination scheme. Additionally, the IES model was implemented on a parallel Beowulf computer cluster which significantly decreased execution time as compared to IES implemented on a regular serial computer.

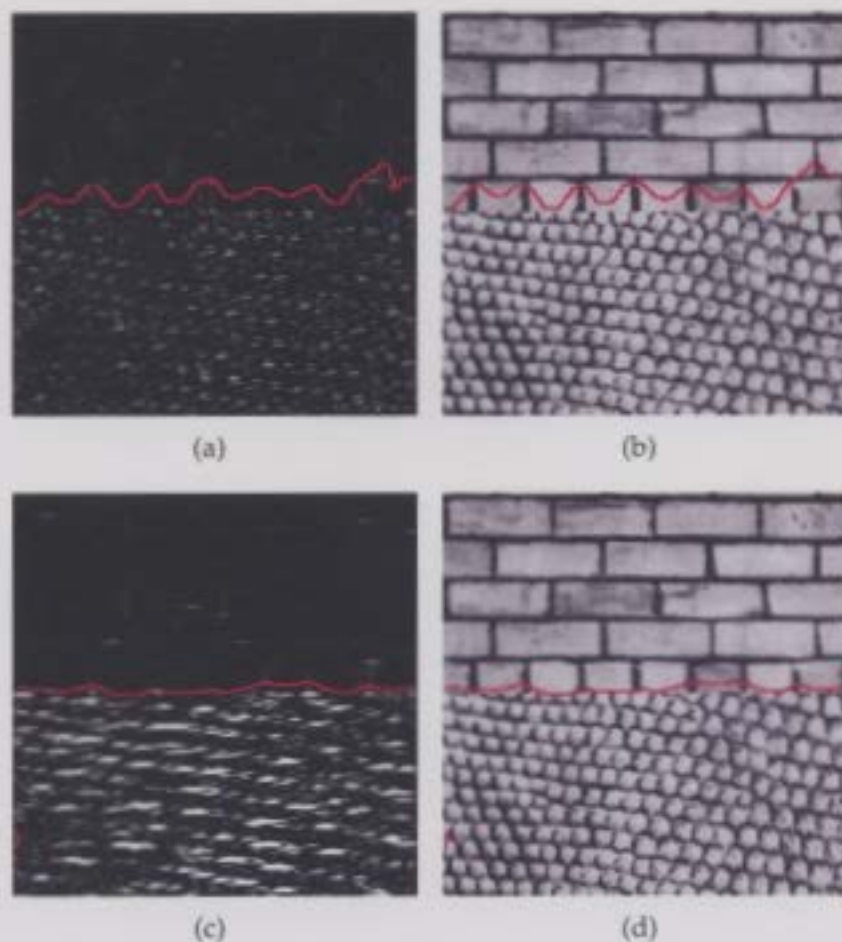


Figure 3.22: (a) ACWE Segmentation result for mosaic overlaid on $|g_1|$ obtained via V-O decomposition, (b) Segmentation result overlaid on original mosaic image for V-O decomp., (c) ACWE Seg. result for mosaic overlaid on $|g_1|$ obtained via IES decomposition, and (d) Segmentation result overlaid on original mosaic image for IES decomp.

CHAPTER 4

Simultaneous Decomposition and Discrimination

4.1 Introduction

It is becoming increasingly common for human image processing researchers to propose and implement algorithms that solve more than one problem at the same time, e.g. deblurring and segmentation [24] or regularization and classification [72]. This approach utilizes the synergy which exists between the solutions of these problems, so that information from the solution of one can simplify the solution of the other, and vice versa.

In this chapter, simultaneous textured image decomposition and discrimination is studied; an earlier version of the material was published in [25]. Modifications here generalize the method to textured images with more than 2 different regions, and there are also some modifications in this thesis to fine points of the algorithm.

This work is inspired by the paper of Vese and Osher [2], where auxiliary texture components obtained from image decomposition are used along with the Active Contours without Edges (ACWE) scheme of Chan and Vese [3] to perform texture discrimination. Here, instead of first performing decomposition and then discrimination, a method is described which alternates between the two individual solutions.

Note that similar work has been published very recently by Bresson and Thiran in [73]. However, their model only uses information from a rough image decomposition to help in the solution of the general segmentation problem, and the information from image segmentation is not utilized for decomposition. Their algorithm uses the Mumford-Shah segmentation model, where within each region, the structure (cartoon) component is assumed to be smooth. On one hand, this approach is better than one based on ACWE, because any number of regions can be segmented with just two level set functions, as can be proven by the Four-Colour Theorem from graph theory [74]. With the ACWE model, in general $\lceil \log_2(n) \rceil$ level set functions are required for n segments. For example, if there are 2 level set functions ϕ_1 and ϕ_2 being used, then the final segmentation of the image I with domain Ω would consist of the regions $\{R_i\}_{i=1}^4$, where

$$\begin{aligned} R_1 &= \{(x, y) \in \Omega \mid \phi_1(x, y) > 0, \phi_2(x, y) > 0\}, \\ R_2 &= \{(x, y) \in \Omega \mid \phi_1(x, y) > 0, \phi_2(x, y) \leq 0\}, \\ R_3 &= \{(x, y) \in \Omega \mid \phi_1(x, y) \leq 0, \phi_2(x, y) > 0\}, \\ \text{and } R_4 &= \{(x, y) \in \Omega \mid \phi_1(x, y) \leq 0, \phi_2(x, y) \leq 0\} \end{aligned}$$

On the other hand, their algorithm doesn't handle large-scale texture, as does the scheme proposed here. Each texton, or repeating texture unit, is considered to be a separate region with Bresson and Thiran's model if the texton is too large, and if they are not considered as textons, but instead as one texture, in the cartoon component, the model performs a smoothing. This is usually alright, but for example in the test image shown later in Figure 4.3(a), the mortar between the bricks would be blurred, which is not desirable, since these should be cartoon edges, by virtue of separating the large-scale bricks in the image. Additionally, the algorithm proposed here makes use of the g_1 and g_2 texture subcomponents in the V-O decomposition model, which leads to efficient segmentation. A combined active contour algorithm is employed, using both local edge information and global region information (without edges). Finally, Bresson and Thiran's method cannot segment two textures of the same mean because the segmentation is made primarily on the basis of the structural component u .

4.2 Proposed Energy Functional

To implement simultaneous textured image decomposition and discrimination, five new terms are added to the sum of the energy $E_{VO}(u, g_1, g_2)$ of Vese and Osher [2], and the energy $E_{ACWE_vec}(\phi, \{u_{0,i}\}_{i=1}^n)$ from the Active Contours without Edges for Vector-Valued images model of Chan and Vese [3]. Recall that the V-O

energy functional is given, as in Equation 3.3.1, by

$$\begin{aligned} E_{\mathbf{VO}}(u, g_1, g_2) = \int_{\Omega} |\nabla u| dx dy + \lambda \int_{\Omega} (f - u - \partial_x g_1 - \partial_y g_2)^2 dx dy \\ + \mu \left[\int_{\Omega} (\sqrt{g_1^2 + g_2^2})^p dx dy \right]^{\frac{1}{p}}, \end{aligned} \quad (4.2.1)$$

The ACWE for Vector-Valued Images energy is based on the ACWE energy in Equation 2.6.2, except that instead of including terms measuring the deviation of the base image from the mean in each region in the energy, the sum of deviations from each of the channels of the base image to each channel mean is included. The energy for ACWE for Vector-Valued images, $E_{\text{ACWE_vec}}$, is given by the equation

$$\begin{aligned} E_{\text{ACWE_vec}}(\phi, \{u_{0,i}\}_{i=1}^n) = \mu_{\text{ACWE}} \int_{\Omega} \delta(\phi(x, y)) |\nabla \phi(x, y)| dx dy + \\ \sum_{i=1}^n \left(\lambda_i^+ \int_{\Omega} (u_{0,i}(x, y) - c_i^+)^2 H(\phi(x, y)) dx dy + \right. \\ \left. \lambda_i^- \int_{\Omega} (u_{0,i}(x, y) - c_i^-)^2 (1 - H(\phi(x, y))) dx dy \right). \end{aligned}$$

where n is the number of channels used for segmentation. The new energy for the proposed SDD scheme, $E_{\text{SDD}}(u, g_1, g_2, \phi)$, as initially published in [25], has the

following form

$$\begin{aligned}
\mathbf{E}_{\text{SDD}}(u, g_1, g_2, \phi) &= \mathbf{E}_{\text{VO}}(u, g_1, g_2) + \mathbf{E}_{\text{ACWE_vec}}(\phi, CS_g|g_i|, CS_s scale_1) \\
&+ \underbrace{\gamma_1 \cdot scale_1 \int_{\Omega} e^{-\frac{\phi^2}{2}} dx dy}_{\text{Term 1}} + \underbrace{\frac{\gamma_2 \cdot scale_2}{2} \int_{\Omega} (f - u)^2 dx dy}_{\text{Term 2}} + \underbrace{\frac{\gamma_3 \cdot scale_3}{2} \int_{\Omega} |\phi| |\nabla u|^2 dx dy}_{\text{Term 3}} \\
&+ \underbrace{\gamma_4 \int_{\Omega} CS_g |\nabla(G_{\sigma} * |g_i|)| |\phi| dx dy}_{\text{Term 4}} + \underbrace{\frac{\mu_{woreinit}}{2} \int_{\Omega} (|\nabla \phi| - 1)^2 dx dy}_{\text{Term 5}}
\end{aligned} \tag{4.2.2}$$

The parameters $\gamma_i, 1 \leq i \leq 4$ and $\mu_{woreinit}$ are positive constants, G_{σ} is a Gaussian filter with standard deviation σ and $*$ denotes convolution. The spatially-varying coefficients $scale_i, 1 \leq i \leq 3$ are calculated using the TV-based scale computation of the following section. One of the channels, $CS_g|g_i|$ is a contrast-stretched version of the absolute value of one of the subcomponents g_i of the decomposition, as is described in more detail in Section 4.3. CS_s is also a contrast-stretching factor, this time for the spatially varying $scale_1$; CS_s is set to 1 for all of the two-region simultaneous decompositions/discriminations in this chapter. The value of CS_s was not varied in the experiments for this chapter, and so it is very possible that other values could lead to better discrimination results. The energy \mathbf{E}_{SDD} in Equation 4.2.2 could be modified by changing the $|\phi|$ factors inside each of the integrands of new terms 3 and 4, to $|\text{sign}(\phi)|$, where the sign function is approximated by $2H_{\epsilon}(\phi) - 1$, with $H_{\epsilon}(\phi)$ the regularized Heaviside or step function. To be even more proper and to avoid the non-differentiability of the absolute value function at the origin, it would be better to take $(2H_{\epsilon}(\phi) - 1) \cdot \text{sign}(2H_{\epsilon}(\phi) - 1) = (2H_{\epsilon}(\phi) - 1) \cdot (2H_{\epsilon}(2H_{\epsilon}(\phi) - 1) - 1)$. Neither the sign function nor its regulariza-

tion were used in [25], or here, but could perhaps improve stability.

The Euler-Lagrange equations can be easily obtained from the calculus of variations. Those for the auxiliary functions g_1 and g_2 are the same as those in Equations 2.4.6 and 2.4.7 from the V-O functional, and so are not included here. The other two for u and ϕ , which are obtained directly from the energy in Equation 4.2.2, are

$$u - f + \partial_x g_1 + \partial_y g_2 - \frac{\gamma_2}{2\lambda} \text{scale}_2 \cdot (f - u) = \frac{1}{2\lambda} \text{div} \left(\frac{\nabla u}{|\nabla u|} \right) + \frac{\gamma_3}{2\lambda} \text{scale}_3 |\phi| \Delta u \quad (4.2.3)$$

$$\begin{aligned} \delta_\epsilon(\phi) & \left(\frac{1}{n} \sum_{i=1}^n \lambda_i^+ (u_{0,i} - c_i^+)^2 - \lambda_i^- (u_{0,i} - c_i^-)^2 \right) - \gamma_1 \text{scale}_1 \cdot \phi e^{-\frac{\phi^2}{2}} \\ & + \gamma_3 \cdot \frac{\text{scale}_3}{2} |\nabla u|^2 \text{sign}(\phi) + \gamma_4 C S_g |\nabla(G_\sigma * |g_i|)| \text{sign}(\phi) \\ & = \mu_{\text{woreinit}} \left(\Delta \phi - 2 \text{div} \left(\frac{\nabla \phi}{|\nabla \phi|} \right) \right) + \delta_\epsilon(\phi) \mu_{ACWE} \text{div} \left(\frac{\nabla \phi}{|\nabla \phi|} \right) \quad (4.2.4) \end{aligned}$$

In Equation 4.2.4, observe that there is a term $\gamma_3 \cdot \frac{\text{scale}_3}{2} |\nabla u|^2 \text{sign}(\phi)$. In the early stages of discrimination, $|\nabla u|$ may be high at small-scale texture edges but these initial estimates eventually dissipate in the decomposition process. Where the gradient magnitude of the cartoon component is high, the level set function ϕ is unduly forced close to zero, creating false boundaries at these small-scale texture edges. So instead, the following modification of the term in Equation 4.2.4 is considered

$$\gamma_3 \cdot \text{scale}_3 \sqrt{|\nabla f_{TV}|} \text{sign}(\phi),$$

where f_{TV} is the original image f evolved according to total variation flow (note the square root). This total variation process has to be computed anyway for the calculation of the scale coefficients, so no extra work is needed for this modification of the Euler-Lagrange equation.

The above energy can be generalized to two level set functions ϕ_1 and ϕ_2 instead of just one function ϕ . This can be done to allow for the discrimination of up to 4 regions. The new energy would be

$$\begin{aligned}
& \mathbf{E}_{\text{SDD},2}(u, g_1, g_2, \phi_1, \phi_2) = \mathbf{E}_{\text{VO}}(u, g_1, g_2) \\
& + \mathbf{E}_{\text{ACWE}_{\text{vec}}^{\text{mphase}}}(\phi_1, \phi_2, CS_g|g_i|, CS_s scale_1, CS_f f_{TV}) \\
& + \gamma_1 \cdot scale_1 \int_{\Omega} \left(e^{-\frac{\phi_1^2}{2}} + e^{-\frac{\phi_2^2}{2}} \right) dx dy + \frac{\gamma_2 \cdot scale_2}{2} \int_{\Omega} (f - u)^2 dx dy + \\
& \frac{\gamma_3 \cdot scale_3}{2} \int_{\Omega} |\phi_1| |\phi_2| |\nabla u|^2 dx dy + \gamma_4 \int_{\Omega} CS_g |\nabla (G_{\sigma} * |g_i|)| |\phi_1| |\phi_2| dx dy + \\
& \frac{\mu_{\text{woreinit}}}{2} \int_{\Omega} [(|\nabla \phi_1| - 1)^2 + (|\nabla \phi_2| - 1)^2] dx dy
\end{aligned}$$

The first new term with coefficient γ_1 has been changed so that in large scale regions, both ϕ_1 and ϕ_2 should be large, to ensure that there are no boundaries for either level set function in such regions. The third new term with coefficient γ_3 has been modified so that if $|\nabla u|^2$ is large, then either there can be a boundary in ϕ_1 or ϕ_2 , but not necessarily both (one of these level set functions ϕ_1 or ϕ_2 should be close to zero). A similar change has been made to the fourth new term with coefficient γ_4 , and finally the last new term ensures that both ϕ_1 and ϕ_2 stay close to a signed distance function. The energy functional $\mathbf{E}_{\text{ACWE}_{\text{vec}}^{\text{mphase}}}$ refers to the energy for ACWE with 2 level set functions ϕ_1 and ϕ_2 (thus the mphase, for multiphase), and has the subscript vec (for vector) because there is more than one channel being segmented. In fact, there are three: $CS_g|g_i|$, $CS_f f_{TV}$ and $CS_s scale_1$. Here, CS_f is a positive contrast-stretching factor, similar to CS_g and CS_s .

For the purposes of stability, it was found that the terms with coefficient γ_3 and γ_4 could be combined. This is because often edges in $|g_i|$ and $|\nabla u|$ (or $|\nabla f_{TV}|$)

coincide. So instead of including

$$\begin{aligned} & \frac{\gamma_3 \cdot scale_3}{2} \int_{\Omega} |\phi_1| |\phi_2| |\nabla f_{TV}| dx dy + \gamma_4 \int_{\Omega} CS_g |\nabla(G_{\sigma} * |g_i|)| |\phi_1| |\phi_2| dx dy \\ &= \int_{\Omega} \left(\frac{\gamma_3 \cdot scale_3}{2} |\nabla f_{TV}| + \gamma_4 |CS_g| |\nabla(G_{\sigma} * |g_i|)| \right) |\phi_1| |\phi_2| dx dy, \end{aligned} \quad (4.2.5)$$

these two terms are changed to

$$\int_{\Omega} \max(\gamma_3 \cdot scale_3 |\nabla f_{TV}|, \gamma_4 |CS_g| |\nabla(G_{\sigma} * |g_i|)|) |\phi_1| |\phi_2| dx dy. \quad (4.2.6)$$

Note that as opposed to the two-region case, $|\nabla f_{TV}|$ was used in Equations 4.2.5 and 4.2.6 instead of $|\nabla u|$. Not much sensitivity was found to this choice, and this could be the subject of further investigation. The change from Equation 4.2.5 to 4.2.6 is done because often edges in $|g_i|$ and $|f_{TV}|$ coincide, so that if these two terms are summed, the contribution of the term to the overall energy will be too great. This was found to be the case experimentally, where the effect of the other terms in the energy functional of Equation 4.2.5 was found to be negligible, though sometimes these terms were necessary to yield an accurate segmentation result.

In Section 4.2.3, it will be seen that the spatially-varying scale coefficients $scale_1$ and $scale_2$ for both E_{SDD} and $E_{SDD,2}$, though computed in slightly different manners, are large in large-scale regions, while $scale_3$ is large in regions with small local scale.

Now the meaning of the new terms in E_{SDD} , and by extension $E_{SDD,2}$ are further explained, keeping in mind the just mentioned behaviour of the scale coefficients.

4.2.1 Meaning of New Terms

Here the purposes for adding each of the five new terms to E_{VO} and E_{ACWE} to obtain E_{SDD} are expounded, one by one:

1. The first additional term, with coefficient γ_1 steers the value of ϕ away from zero in regions of large-scale. This is done because, in general, there will not be texture boundaries in such large-scale regions. If an image is piecewise-smooth and doesn't contain textures, then the other terms in the energy will override this term. This term can also be turned off by setting γ_1 to zero, since other more straightforward methods such as edge detection will work. An exponential is used in the integrand because unlike $-\phi^2$, $e^{-\frac{\phi^2}{2}}$ is bounded below, while still being decreasing in ϕ .
2. The second additional term is a fidelity term (the cartoon component is kept close to the original image f) which ensures that large-scale features are not placed in the texture component v or the residual ($r = f - u - v$). This is accomplished only for large-scale regions because the coefficient includes a multiplicative factor of $scale_2$ which is only high in such regions.
3. The third extra term ensures that in small-scale regions, $|\nabla u|^2$ is small when $|\phi|$ is large. This term is only turned on in small-scale regions because of the $scale_3$ factor in the coefficient. So in small-scale regions, far away from the texture boundaries, blurring of the cartoon component u is promoted. This is an example of how the discrimination process helps decomposition.
4. In order to reduce the number of iterations necessary for ACWE to converge

to the correct discrimination result, some local edge information was also included for that technique. This is similar in spirit to what Kimmel and Bruckstein [75] do except that only the gradient magnitude is considered instead of the gradient vector, and here a Gaussian-blurred version is used in the proposed algorithm. The term with coefficient γ_4 accomplishes this by forcing ϕ to be small in absolute value when this gradient is large. Thus, when the gradient is large, there will be a boundary since the level set boundary is defined implicitly by $\phi = 0$. Unlike Kimmel and Bruckstein's extra term, this term is robust to noise because of the Gaussian blurring.

5. The final term is taken from the paper by Li et. al. [76]. It provides an alternative to having to reinitialize the level set function every several iterations. Instead the level set function ϕ is kept close to a signed distance function by keeping the difference between the norm of its gradient and unity small. This is based on the easily seen fact that the gradient magnitude of any function h is equal to the magnitude of the maximum directional derivative at that point. This fact will now be shown. The maximal directional derivative of the function h occurs in a direction perpendicular to the level set at that point, and is given by the formula $h(x, y) = (\partial_x h, \partial_y h)$. Locally, if any point z on C is chosen as the origin, the distance function on one side of the curve to z will equal $\phi(x, y) = \sqrt{x^2 + y^2}$, and so the gradient magnitude will equal $|\nabla \phi| = \left| \left(\frac{x}{\sqrt{x^2 + y^2}}, \frac{y}{\sqrt{x^2 + y^2}} \right) \right| = 1$, as desired. This term stabilizes the evolution of the level set, keeping the level set function smaller in magnitude than would be the case without the term.

The calculation of the scale coefficients $scale_1$, $scale_2$ and $scale_3$ for E_{SDD} and its generalization $E_{SDD,2}$ is now described.

4.2.2 Calculation of Scale

Brox and Weickert [77] determined the scales of textures in an image using the rate of change of the gray-levels of pixels when the image underwent total variation (TV) flow. This underlying flow can either be implemented semi-implicitly and efficiently with Additive Operator Splitting (AOS, [36]), or explicitly and less efficiently with forward time-stepping.

There are two underlying formulae valid for both approaches. One formula contains an explicit stopping time T , whilst the other has no stopping parameter [77].

These two formulae are

$$scale_{Brox,stop} = 4 \frac{\int_0^T (1 - \delta_{\partial_t u, 0}) dt}{\int_0^T |\partial_t u| dt} \quad (4.2.7)$$

$$scale_{Brox,nostop} = 4 \frac{\int_0^{T_{\max}} |\partial_t u| dt}{\int_0^{T_{\max}} |\partial_t u|^2 dt} \quad (4.2.8)$$

where u is evolved with a TV flow from time 0 to T . The function $\delta_{x,y}$ equals 1 if $x = y$ and zero otherwise. In the second formula, without a stopping time, T_{\max} is the time when the image reaches equilibrium, or the time at which the estimate doesn't change more than a certain small ϵ in L^2 -norm, between iterations. As usual, $|\partial_t u|$ refers to the absolute value of the change in u from one iteration to the next.

It was found that by using AOS for the TV flow, large regions had bigger scale, but their boundaries did not. By using explicit forward time-stepping for the TV flow, object boundaries had big scale while the scale determined for large regions was not as big. One option is to linearly combine the two scale measurements obtained with the AOS ($scale_{AOS}$) and explicit ($scale_{exp}$) TV flows. Instead, reasoning based on mathematical morphology was used to combine $scale_{AOS}$ and $scale_{exp}$ in a meaningful manner to form the three scale coefficients $scale_1$, $scale_2$ and $scale_3$, given above in Equation 4.2.2. In this manner, $scale_1$ ensures there are no discrimination contours in large-scale regions or small-scale regions within such large-scale regions, the coefficient $scale_2$ preserves the cartoon component in large-scale regions, and $scale_3$ promotes blurring of the cartoon component in small-scale regions and promotes the presence of discrimination contours at cartoon edges that remain in such small-scale regions. Since it did not require much extra time, the scale formula with no fixed stop time was used for $scale_{AOS}$. Explicit time-stepping took longer to compute, so the formula with stopping time from [77] and Equation 4.2.8 was used for $scale_{exp}$.

Scale itself can be used as a cue for texture discrimination. To include this important information, ACWE for vector-valued images [53] is used in the proposed SDD algorithm, with the data vector consisting of the absolute value of g_i and $scale_1$.

4.2.3 Calculation of Coefficients based on Scale

The scale coefficient $scale_1$ of the first term will be high where it is known there are no discrimination contours. This is found by either taking a morphologically closed (with a flat square structuring element of size 5) $scale_{AOS}$ when the distance to a pixel with high $scale_{exp}$ value (greater than 30) is less than or equal to 3, or $scale_{AOS}$ otherwise.

The second scale-based coefficient $scale_2$ is calculated by conditionally dilating $scale_{AOS}$ with the same type of structuring element. This dilation is initially done unconditionally, and then for each pixel the value of $scale_2$ is set to be equal to that obtained after dilation if the pixel is within a Euclidean distance of 3 from a pixel with high $scale_{exp}$ (again greater than 30), and equal to $scale_{AOS}$ otherwise.

Finally, $scale_3$ is meant to be large in small-scale regions. The term $\frac{1}{scale_1+0.1}$ (the 0.1 in the denominator added to avoid division by 0) is added to $\frac{5}{G_{\sigma_{TV}} * |\nabla f_{TV}| + 1}$ where $\sigma_{TV} = 5$, with the two summands normalized so their maxima are equal. In this expression f_{TV} is the original image having undergone TV-flow with AOS nonlinear diffusion. Only pixels in regions of size greater than 250 where the first summand is greater than 0.25 were considered, to eliminate spurious small regions.

4.3 Contrast Stretching

Especially in the early stages of decomposition when using the same initialization as in [2], $u = f$, the absolute values of the auxiliary texture components g_1 and

g_2 are small. In order to magnify the differences in $|g_1|$ and $|g_2|$ in the first iterations, contrast stretching is performed by multiplying the component by a scalar iteration-dependent factor (> 1).

This factor is given by

$$CS_g = \frac{20 + (n_{totaldec}/10 - 1)}{2 + (n_{totaldec}/10 - 1)} \quad (4.3.1)$$

where $n_{totaldec}$ is the total number of decomposition iterations that have been run. This number is necessarily less than or equal to 10, because decomposition is initially run for 10 iterations, and CS_g is only used for discrimination, which starts after the initial decomposition iterations. A plot of how the contrast-stretching factor varies with iteration is shown in Figure 4.1.

Different forms of the contrast stretching factor CS_g could be considered. As decomposition progresses, both the g_1 and g_2 subcomponents increase in their magnitude. Therefore, it is reasonable to have a contrast stretching factor which is decreasing as the number of decomposition iterations, $n_{totaldec}$, increases, since otherwise the terms in $E_{SD}(u, g_1, g_2, \phi)$ including this CS_g factor would have too much influence in the overall energy. The form of CS_g in Equation 4.3.1 was the first attempted, with only experimentation performed on the added term 20 in the numerator and the added term 2 in the denominator of this equation.

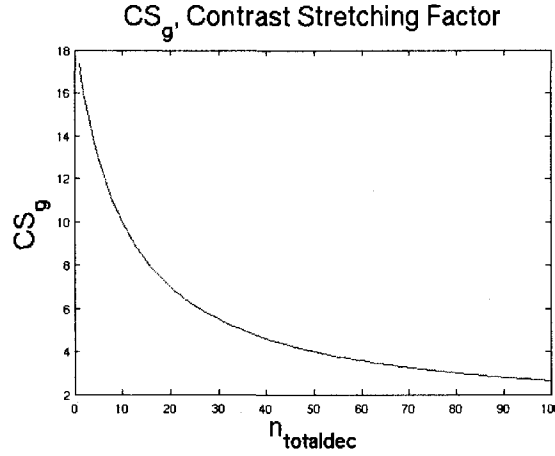


Figure 4.1: Iteration-Dependent Change in Contrast-Stretching Factor CS_g

4.4 Methodology

4.4.1 Numerical Implementation

The alternation between decomposition and discrimination is described by the flow diagram in Figure 4.2 and the pseudocode in Algorithm 2. In the flow diagram of Figure 4.2, the statements in each rectangle are executed while the program is in that state, and solid arrows are followed if the condition written next to them holds true. If there is no condition written next to a solid arrow, then the solid arrow is followed unconditionally after the statements in the current rectangle have been executed. Dotted arrows represent data flow, so that variables from the start of the arrow are fed into the state at the arrow's end.

The variables `decompiter` and `discriminator` represent the number of iterations of decomposition and discrimination respectively and were chosen in an image-dependent manner.

Vese and Osher's (u, v) decomposition is implemented in their paper with a

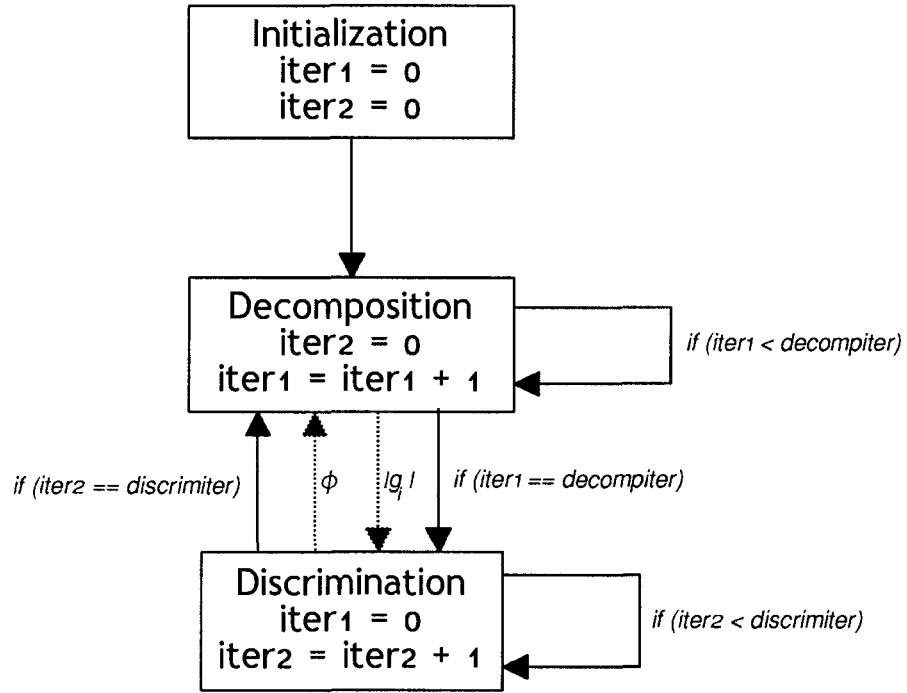


Figure 4.2: Flow Diagram for SDD Algorithm

Algorithm 2 Pseudocode for Simultaneous Decomposition/Discrimination

```

1: procedure INITIALIZATION( $u, g_1, g_2, \phi$ )
2:    $u = u^0, g_1 = g_1^0, g_2 = g_2^0$  and  $\phi = \phi^0$ ;
3:    $j = 0$ ;
4:    $k = 0$ ;
5: end procedure
6: procedure MAIN SDD ALGORITHM( $u, g_1, g_2, \phi$ )
7:   while at least one of  $u, g_1, g_2$  or  $\phi$  has not converged do
8:     for  $i \leftarrow 1, decompiter$  do
9:        $(u^{j+1}, g_1^{j+1}, g_2^{j+1}) = decomp((u^j, g_1^j, g_2^j), \phi^k)$ ;
10:       $j = j + 1$ ;
11:    end for
12:    for  $i \leftarrow 1, discriminator$  do
13:       $\phi^{k+1} = discrim(\phi^k, (u^j, g_1^j, g_2^j))$ ;
14:       $k = k + 1$ ;
15:    end for
16:  end while
17:   $u = u^j, g_1 = g_1^j, g_2 = g_2^j, \phi = \phi^k$ ;
18: end procedure

```

semi-implicit fixed-point iterative finite differences scheme, based on that of Aubert and Vese [45]. A similar discretization is followed for the minimization of the proposed functional for decomposition.

For the active contour evolution, a discretization similar to that found in the literature [3] is used, except that a semi-implicit Gauss-Seidel procedure was followed like that of the decomposition procedure, instead of a fully implicit one. The time step used for the level set discretization is $\Delta t = 0.1$.

The initial conditions for the components/subcomponents of the image decomposition are $u_0 = f$, $g_1 = -\frac{1}{2\lambda} \frac{f_x}{|\nabla f|}$ and $g_2 = -\frac{1}{2\lambda} \frac{f_y}{|\nabla f|}$, as were used for V-O decomposition [2]. The initial condition for ϕ is a set of horizontally and vertically aligned small circles as shown in Figure 2.5 of Chapter 2.

The defining equation for ϕ for normal Active Contours Without Edges [3] at the $(n + 1)^{st}$ iteration given the values of ϕ at the n^{th} iteration is given by

$$\begin{aligned} \phi_{i,j}^{n+1} = & \frac{\phi_{i,j}^n + \Delta t \delta_h(\phi_{i,j}^n) \left[\frac{\mu}{h^2} (c_1 \phi_{i+1,j}^n + c_2 \phi_{i-1,j}^{n+1} + c_3 \phi_{i,j+1}^n + c_4 \phi_{i,j-1}^{n+1}) \right]}{1 + \Delta t \delta_h(\phi_{i,j}^n) \left[\frac{\mu}{h^2} (c_1 + c_2 + c_3 + c_4) \right]} \\ & + \frac{\Delta t \delta_h(\phi_{i,j}^n) [-\lambda_1 (u_{0,i,j} - m_1(\phi^n))^2 + \lambda_2 (u_{0,i,j} - m_2(\phi^n))^2]}{1 + \Delta t \delta_h(\phi_{i,j}^n) \left[\frac{\mu}{h^2} (c_1 + c_2 + c_3 + c_4) \right]}, \end{aligned} \quad (4.4.1)$$

where

$$\begin{aligned}
c_1 &= \frac{1}{\sqrt{(\frac{\phi_{i+1,j}^n - \phi_{i,j}^n}{h})^2 + (\frac{\phi_{i,j+1}^n - \phi_{i,j-1}^n}{2h})^2}}, \\
c_2 &= \frac{1}{\sqrt{(\frac{\phi_{i,j}^n - \phi_{i-1,j}^n}{h})^2 + (\frac{\phi_{i-1,j+1}^n - \phi_{i-1,j-1}^n}{2h})^2}}, \\
c_3 &= \frac{1}{\sqrt{(\frac{\phi_{i+1,j}^n - \phi_{i-1,j}^n}{2h})^2 + (\frac{\phi_{i,j+1}^n - \phi_{i,j}^n}{2h})^2}} \text{ and} \\
c_4 &= \frac{1}{\sqrt{(\frac{\phi_{i+1,j-1}^n - \phi_{i-1,j-1}^n}{2h})^2 + (\frac{\phi_{i,j}^n - \phi_{i,j-1}^n}{h})^2}}.
\end{aligned}$$

This discretization can be extended easily to the level set iterative solution of the simultaneous decomposition/discrimination scheme. In these equations h is the grid spacing which can be taken to be unity, without loss of generality.

4.4.2 Varying the coefficients specific to the algorithm

The coefficients $\gamma_i, 1 \leq i \leq 4$, of the new terms can be varied depending on whether decomposition or discrimination is being performed, since at any instance, only iterations for one of the algorithms is being run. It was found by experiment that as long as these coefficients are kept constant within each algorithm, the entire simultaneous decomposition and discrimination (SDD) algorithm gives meaningful results.

The solution thus found by the entire process is not likely to be a global minimum of the energy $E_{\text{SDD}}(u, g_1, g_2, \phi)$, but still yields better results than either of the algorithms run separately, as will be illustrated in Section 4.5.1. This is only an

issue for the coefficient parameters γ_3 and γ_4 since these are the only coefficients of the integrals in E_{SDP} which depend both on an element of $\{u, g_1, g_2\}$ and ϕ .

4.4.3 Test Images

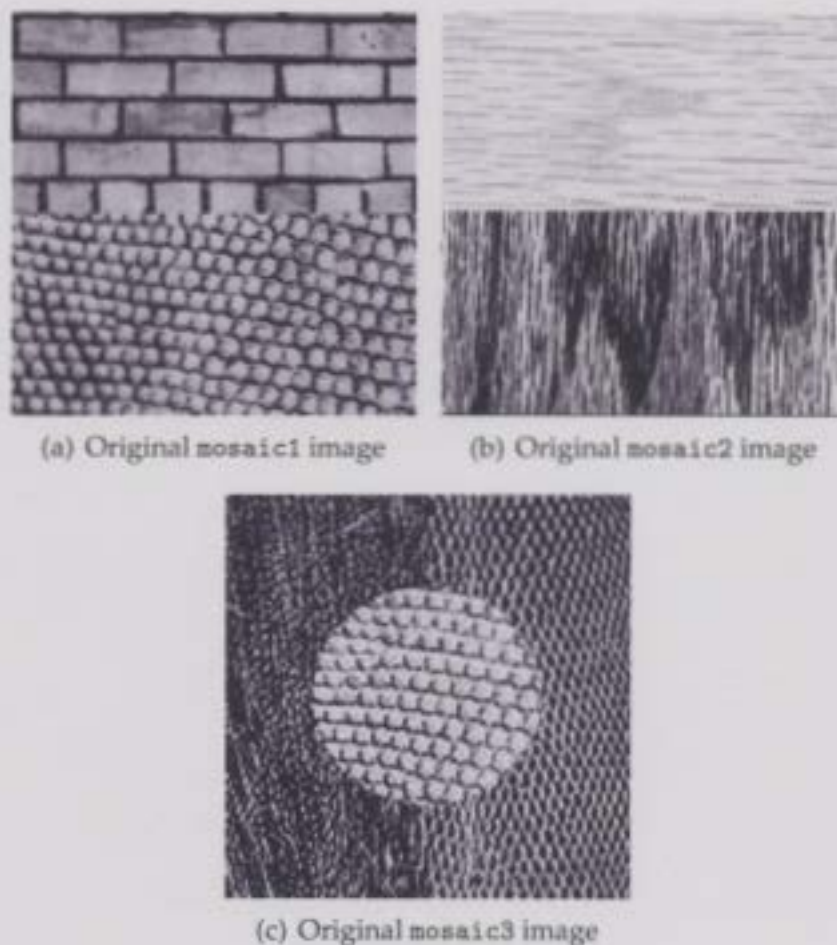


Figure 4.3: Test Images used in Experiments

Three test images used in the experiments in this chapter, *mosaic1*, *mosaic2* and *mosaic3*, are shown in Figure 4.3 below. The first two images are 160x160 pixels large, while the third is 192x192 pixels in size. Note the need for at least two level set functions for proper discrimination of *mosaic3* (because it contains 3 different

textures) motivates its inclusion in the test set.

4.4.4 Parameter Selection

Image name	μ	λ	γ_1	γ_2	$\gamma_{3,dp}$	$\gamma_{3,dsc}$
mosaic1	0.05	0.02	50	25	2.5	30
mosaic2	0.05	0.005	30	30	50	300
mosaic3	0.05	0.01	0	0	0.01	5

Table 4.1: Parameter choices for SDD on test images

Image name	$\gamma_{4,dp}$	$\gamma_{4,dsc}$	$\mu_{woreinit}$	$\mu_{ACWE} (\times 255^2)$	decompiter	discrimiter
mosaic1	0	50	20	1.0	10	50
mosaic2	0	70	70	2.0	20	110
mosaic3	0	5	20	3.0	10	150

Table 4.2: Parameter choices for SDD on test images (cont.)

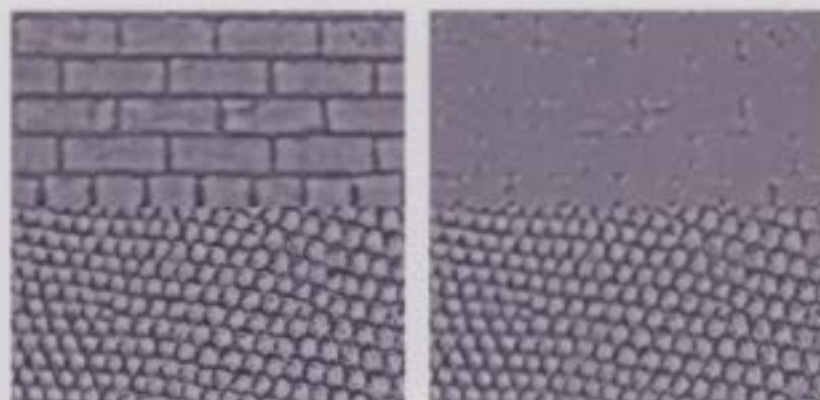
There are many parameters, e.g. coefficients of terms in energy functionals, that should be selected for the proposed SDD algorithm. The values of these parameters for the images mosaic1, mosaic2 and mosaic3 are shown in Tables 4.1 and 4.2. These choices were not optimized - it is very likely that there are better values of the parameters, e.g. for decompiter and discrimiter, that would lead to quicker convergence, or slightly better solutions in quality. The decomposition parameters μ and λ for Vese-Osher decomposition used in sequential decomposition/discrimination are the same as those for SDD. The grouping parameter μ_{ACWE} for the discrimination part of the sequential scheme is set to 0.2×255^2 both for mosaic1 and mosaic2. The grouping parameter μ_{ACWE} is different for SDD on the three images: 1.0×255^2 for mosaic1, 2.0×255^2 for mosaic2 and 3.0×255^2 for mosaic3. In Active Contours without Edges for Vector-Valued images, λ_1^\pm and λ_2^\pm are set to 1, both for the minimization of E_{SDD} and $E_{SDD,2}$.

4.5 Results and Discussion

4.5.1 Two-Region Test Images



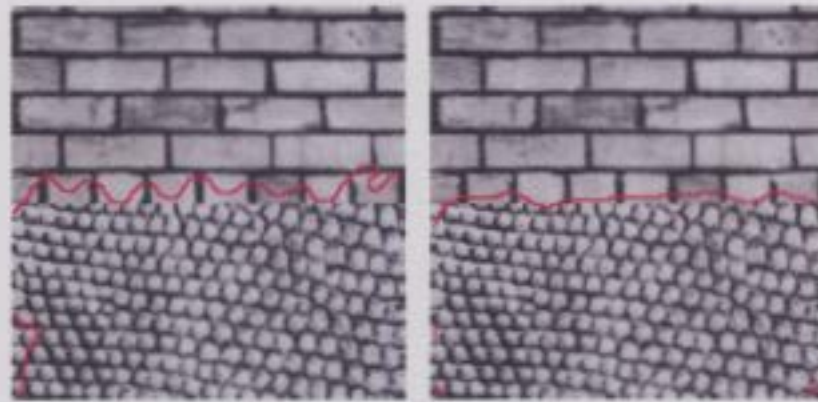
(a) Cartoon component u of mosaic1 with V-O decomp. (b) Cartoon component u of mosaic1 with SDD



(c) Texture component v of mosaic1 with V-O decomp. (d) Texture component v of mosaic1 with SDD

Figure 4.4: Image decomposition results from V-O and SDD decomposition for mosaic1

Decomposition results for the image mosaic1 are shown in Figure 4.4. Discrimination results (computed via the $|g_1|$ component of the same image) are given in Figure 4.5. As can be seen by comparing Figures 4.4(a) and 4.4(b), the large-scale brick texture in the top half of mosaic1 is preserved almost perfectly in the cartoon



(a) ACWE discrimination result overlaid on original image (b) ACWE discrimination result overlaid on original image after SDD decomposition

Figure 4.5: Texture discrimination results from V-O and SDD decomposition for `mosaic1`

component for SDD, but for V-O decomposition this component and its cartoon edges are blurred considerably, which is undesirable [40]. More remarkable is that this is accomplished in only 20 decomposition iterations in the proposed scheme instead of 101 for V-O decomposition.

In Figure 4.5, it can be observed that the contour between the two regions is more accurate with SDD rather than with ACWE performed sequentially after decomposition. What is notable, is that as seen in Table 4.3, only 100 iterations of ACWE discrimination are needed for SDD, as opposed to 220 for ACWE in a sequential framework.

It is more difficult to decompose and discriminate the textures of the test image `mosaic2` than of `mosaic1` because there are cartoon edges within the texture, and the texture is very sparse. Even though there are no longer any large-scale structures to preserve, there are still comparable decomposition results with fewer iterations (30 iterations for SDD vs. 80 for sequential decomposition/discrimina-

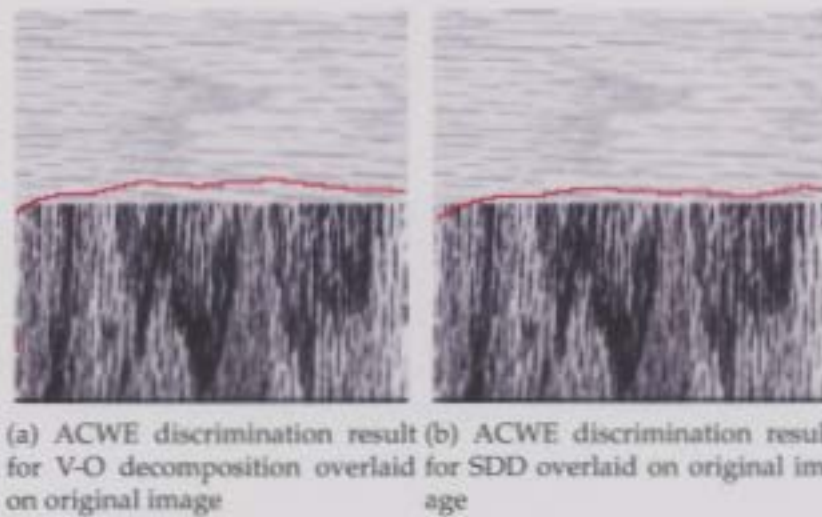


Figure 4.6: Texture discrimination results from V-O and SDD decomposition for mosaic2

tion). Once again $|g_1|$ is used for discrimination, and the discrimination results for both the sequential and simultaneous schemes are shown in Figure 4.6. Both schemes require 220 iterations for discrimination, though with adjustment of the parameters, it is believed that the performance of the simultaneous scheme could be improved. Though of acceptable quality, improving decomposition results such as in Figure 4.7 is left for future work.

Image	Sequential		Simultaneous	
	n_{dcmp}^{VO}	n_{ACWE}	n_{dcmp}^{SDD}	n_{ACWE}^{SDD}
mosaic1	101	220	20	100
mosaic2	80	220	30	220

Table 4.3: Number of iterations required for SDD, and V-O decomposition sequentially followed by ACWE discrimination on two-region test images

In Table 4.3, the number of iterations required for the various components of both sequential decomposition/discrimination and simultaneous decomposition/discrimination (SDD) are given. It can be seen that the number of iterations required both for decomposition and discrimination does not increase for SDD from the se-



(a) Cartoon component u of mosaic2 with V-O decomp. (b) Cartoon component u of mosaic2 with SDD

Figure 4.7: Image decomposition results from V-O and SDD decomposition for mosaic2

Image	Sequential			Simultaneous			
	t_{VO}^{seq}	t_{ACWE}^{seq}	t_{seq}	t_{scale}^{SDD}	t_{dcmp}^{SDD}	t_{ACWE}^{SDD}	t_{SDD}
mosaic1	45.92	58.03	103.95	17.34	76.30	30.67	124.31
mosaic2	32.78	58.09	90.87	10.95	113.78	85.06	209.79
mosaic3	-	-	-	19.70	172.86	159.88	352.44

Table 4.4: Time (secs) required for SDD and V-O decomposition sequentially followed by ACWE discrimination on two and three-region test images

quential scheme. In Table 4.4, the time required for the proposed simultaneous method vs. the sequential decomposition/discrimination on the test images are shown. The experiments were performed on a 2.6 GHz Pentium 4 desktop with 1 GB of RAM and a MATLAB implementation. The column with t_{VO}^{seq} shows the time necessary for Vese-Osher decomposition, and t_{ACWE}^{seq} is the time necessary for ACWE segmentation after V-O decomposition has been run. The column t_{seq} refers to the time necessary for the entire sequential decomposition/discrimination scheme, and so equals $t_{VO}^{seq} + t_{ACWE}^{seq}$. The quantity t_{scale}^{SDD} refers to the amount of time required for pre-computation of the various scale coefficients for SDD in MATLAB, the column t_{dcmp}^{SDD} to the total amount of time necessary for the decom-

position and t_{ACWE}^{SDD} the total time necessary for discrimination, all in the proposed SDD scheme. The column $t_{SDD}(= t_{scale}^{SDD} + t_{decomp}^{SDD} + t_{discrim}^{SDD})$ refers to the total time necessary for SDD to run on each test image. As can be seen from Table 4.4, the time required for the complete SDD process is slightly more than the sequential process for the mosaic1 test image. However, SDD gives both superior decomposition and discrimination results. For mosaic2, SDD takes more than double the time, due in large part to the fact that the entire image is mostly of one scale. The simultaneous scheme is not optimized for such images, though it does give good quality results, as evidenced by the ultimate discrimination results in Figure 4.6.

4.5.2 Test Image with More Than Two Regions

As is obvious from Figure 4.3, mosaic3 has three distinct textured regions to be discriminated. The Euler-Lagrange equations derived from the energy functional $E_{SDD,2}$ in Equation 4.2.5 are used to simultaneously decompose/discriminate this image. It would be expected that even though there are now two level set functions which could handle up to four distinct regions, the problem of discrimination in this case will be more difficult than the two-region case, because the three regions have to be in general pair-wise discriminated from each other.

Because of this extra difficulty, for the test image mosaic3, an extra channel, f_{TV} is fed in to the ACWE for Vector-Valued Images discrimination process. The terms with coefficients γ_3 and γ_4 from Equation 4.2.5 are also combined in the discrimination process, because both are quite similar. Recall that for the two-region case, $\sqrt{|\nabla f_{TV}|}$ was used in the place of $|\nabla u|^2$. A similar replacement, this time with

$|\nabla f_{TV}|$ is done for the multiphase case with 2 level set functions ϕ_1 and ϕ_2 . For the 3-region case on `mosaic3`, the absolute value of the g_2 texture subcomponent is used for discrimination.

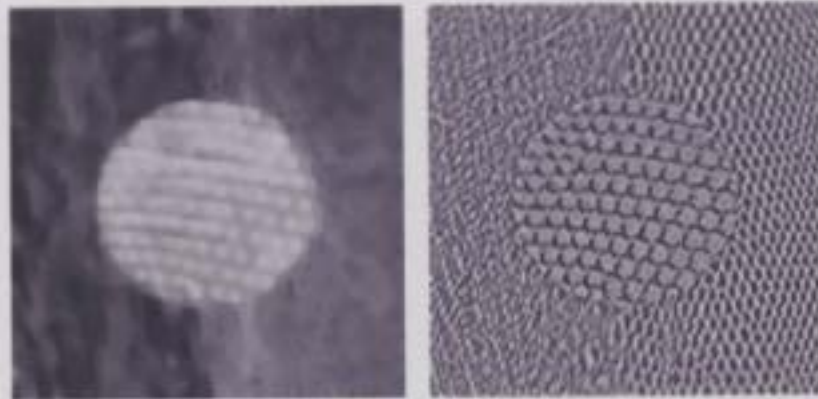
The expression for the iteration-dependent contrast stretching parameter CS_g is slightly different than that for the two-region case in Equation 4.3.1, but is of the same form. The new expression is

$$CS_g = \frac{50 + 3(n_{totaldec}/10 - 1)}{2 + 3(n_{totaldec}/10 - 1)} \quad (4.5.1)$$

The two contrast stretching factors CS_f and CS_s are chosen so that the maximum value of channels $CS_f f_{TV}$ and $CS_s scale_1$ for the multiphase ACWE discrimination are 0.5 and 0.25 times that of $CS_g |g_2|$. It was found important that the value of CS_f be chosen carefully, because if too large or too small, the channel $CS_f f_{TV}$ would have either too much or too little influence on the ACWE discrimination.

It is also important to note that for the multiphase SDD model defined by the energy $E_{SDD,2}$, Jacobi iterations were used instead of Gauss-Seidel iterations to accelerate solution of the model, because with Gauss-Seidel iterations, the solution required substantially more time. This was not done for the two-region case but preliminary experiments with that case showed that the difference in running time between the two implementations was not very great.

The SDD scheme was only run on the test image `mosaic3` to show that the model could be extended to images with more than 2 textured regions. It was found that, in fact, the extension to this case is possible, with a good quality of decomposition and discrimination. Only 30 decomposition iterations and 450 discrimination



(a) Cartoon Component u

(b) Texture Component v

Figure 4.8: Decomposition Result of Proposed SDD Algorithm for mosaic3 image

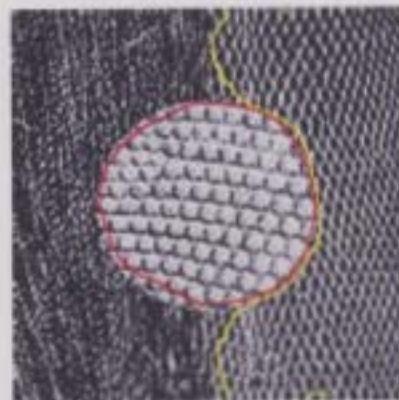


Figure 4.9: Discrimination Result of Proposed SDD Algorithm on mosaic3

iterations were required. The amount of CPU time needed for the entire simultaneous decomposition/discrimination process was only about 6 minutes. Because the total number of level-set functions needed to segment an image with n regions, is only $\lceil \log_2(n) \rceil$, there is not expected to be a substantial decrease in efficiency even for images where more regions to be segmented are present. The sequential decomposition/discrimination algorithm was not run on this image, but this problem is very difficult since there are three different regions, and the cartoon component is not fed into the discrimination part of the sequential algorithm.

4.6 Conclusions

In this chapter, it was shown how decomposition and performing texture discrimination can be naturally solved in conjunction with each other, improving the results obtained by each. The measurement of local scale [77], was a vital part of this new procedure, and was instrumental in preserving the large scale structure in the image, especially in the 2-region case, to improve both decomposition and discrimination results. One problem found in this chapter's experiments was the selection of many different parameters needed for SDD, both for the two-region and multi-region cases. The SDD model was robust to variation in some less important parameters, for example the widths of Gaussian filters used for blurring the scale coefficients, but the rate of convergence was very sensitive to relatively small variations in some of the other parameters, e.g. the contrast-stretching coefficient CS_g . However, overall the SDD model was effective for the simultaneous decomposition and discrimination of textured images with 2 or more regions.

Modifying the Osher-Solé-Vese Decomposition Model

5.1 Introduction

In this chapter, two ways of extending the Osher-Solé-Vese (OSV) image decomposition model are explored. The first introduces a decorrelation term to the energy functional of OSV decomposition based on the correlation coefficient between the cartoon and texture components of the decomposition. For decomposition, improved results over OSV are obtained with the proposed decorrelated model. There are two models which fall under the framework of the second extension, which combines image decomposition with nonlinear diffusion for the purpose of image denoising. The first model combines Perona-Malik diffusion with OSV decomposition, and is abbreviated PMOSV, but unfortunately it does not give very good denoising results. However, the second model, called OLOSV for short, com-

bines oriented Laplacian diffusion and OSV decomposition, and is shown to give dramatically better results for images with oriented texture. The chapter begins, in the next section, with a description of the Decorrelated Osher-Solé-Vese (DOSV) model.

5.2 Decorrelating decomposition components

Although the OSV decomposition model outperforms the V-O model in terms of separating cartoon from texture edges into their respective components [4], resulting decompositions are not always of the highest quality. For example, often cartoon edges appear in the texture component v even though they belong in the cartoon component u . Therefore it is desirable to somehow improve the quality of decompositions obtained from the Osher-Solé-Vese model. A new model is proposed by regularizing that of Osher, Solé and Vese, based on the assumption that the cartoon and texture components of a decomposition are generated from independent processes, and thus are uncorrelated.

In [78], Aujol, et. al. propose using the correlation between the cartoon and texture components of a decomposition to determine the regularization parameter λ by finding a local minimum of this correlation as λ is varied. The assumption made is that the cartoon component u and the texture v are uncorrelated, which could signify that they are generated from independent processes. In practice, they will have some correlation, but this value will be very close to zero.

However, to find this correlation minimum, the method proposed by Aujol, et. al. must evolve the original image with a whole range of λ 's to form a scale space,

a process which requires excessive computation time. Also, the underlying energy functional is not changed. Instead, it is proposed here to impose a regularization term which ensures the correlation between the two components is not significant. Because the residual doesn't come into play, this term is added to a pure two-component model, namely the OSV model, where there are only the cartoon component u , and the texture/noise component $v = f - u$.

The absolute value of the correlation is measured in small windows around each pixel and integrated across the entire image. The implicit assumption is that each small window can be considered to be a random sample of the whole image, and that the independence of the cartoon and texture components holds for this small random sample. As will be seen in the results in Section 5.2.6, while this may be a simplifying assumption, it allows good decomposition results to be obtained.

5.2.1 Correlation coefficient

From statistics, the correlation coefficient between two random variables X and Y is defined to be

$$\rho_{X,Y} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \bar{X})(Y - \bar{Y})]}{\sigma_X \sigma_Y}, \quad (5.2.1)$$

where $E[\cdot]$ (note the square brackets), and an overbar denote the expectation of a random variable, and σ denotes the standard deviation of the random variable appearing in its subscript. This correlation coefficient measures the amount of linear dependence between X and Y , and ranges between -1 and 1. If two random variables are independent, then their correlation coefficient will be zero, but the reverse implication does not necessarily hold.

Recall from Section 2.4.3 that the energy for OSV decomposition can be defined by

$$\mathbf{E}_{\text{OSV}}(u) = \int_{\Omega} |\nabla u| dx dy + \lambda \int_{\Omega} |\nabla(\Delta^{-1})(f - u)|^2 dx dy. \quad (5.2.2)$$

It is shown in [4] that this energy can be minimized by evolving the cartoon component u according to the following partial differential equation

$$u_t = -\frac{1}{2\lambda} \Delta \left[\text{div} \left(\frac{\nabla u}{|\nabla u|} \right) \right] - (u - f). \quad (5.2.3)$$

The right-hand side of the above equation is actually the negative Laplacian of the formally derived first variation of the energy 5.2.2, however in [4], it is shown that this quantity still results in a gradient descent direction of the energy functional of Equation 5.2.2.

To force the correlation between cartoon and texture to be small, the integral term

$$\mathbf{E}_{\text{decorrel}}(u) = \gamma_d \int_{\Omega} |\rho_{u,v}^{W(x,y)}(x, y)| dx dy \quad (5.2.4)$$

is added to the Osher-Solé-Vese energy $\mathbf{E}_{\text{OSV}}(u)$ in Equation 5.2.2, to form the new Decorrelated OSV (DOSV) model energy $\mathbf{E}_{\text{DOSV}}(u)$

$$\begin{aligned} \mathbf{E}_{\text{DOSV}}(u) &= \mathbf{E}_{\text{OSV}}(u) + \mathbf{E}_{\text{decorrel}}(u) \\ &= \int_{\Omega} |\nabla u| dx dy + \lambda \int_{\Omega} |\nabla(\Delta^{-1})(f - u)|^2 dx dy \\ &\quad + \gamma_d \int_{\Omega} |\rho_{u,v}^{W(x,y)}(x, y)| dx dy. \end{aligned} \quad (5.2.5)$$

The correlation coefficient $\rho_{u,v}^{W(x,y)}$ at a pixel (x, y) is taken over a square window

$W(x, y)$ of width w_x and of height w_y about the pixel (x, y) . The formula for this spatially varying correlation coefficient, as obtained from Equation 5.2.1 is

$$\begin{aligned}
 \rho_{u,v}^{W(x,y)} &= \frac{E_{W(x,y)}[(u - \bar{u})(f - u - \overline{(f - u)})]}{\sigma_u \sigma_{f-u}} \\
 &= \frac{\frac{1}{w_x w_y} \int_{W(x,y)} (u - \bar{u})(f - u - \overline{(f - u)}) dx dy}{\sqrt{E_{W(x,y)}[(u - \bar{u})^2] E_{W(x,y)}[(f - u - \overline{(f - u)})^2]}} \\
 &= \frac{\frac{1}{w_x w_y} \int_{W(x,y)} (u - \bar{u})(f - u - \overline{(f - u)}) dx dy}{\frac{1}{w_x w_y} \sqrt{\int_{W(x,y)} (u - \bar{u})^2 dx dy \int_{W(x,y)} (f - u - \overline{(f - u)})^2 dx dy}}
 \end{aligned} \tag{5.2.6}$$

A global correlation coefficient over the entire image is not taken, since there would be a global mean \bar{u} used in the calculation, and any given pixel would contribute directly to this mean. The means should only be taken over a local window, since in general, the average intensities of different objects in the image will not be the same.

It was also attempted to add the term $\int_{\Omega} (\rho_{u,v}^{W(x,y)}(x, y))^2 dx dy$, but this was too permissive of large values of ρ . This extra term is too permissive because the ρ values will necessarily lie between 0 and 1, and by squaring these correlation values in the interval $[0,1]$, larger correlations will still be considered as less significant in the additional term in the energy.

5.2.2 Simplifying Assumptions

To derive the Euler-Lagrange equations of $E_{\text{DOSV}}(u)$, various assumptions on the correlation coefficient field $\rho_{u,v}^{W(x,y)}$ calculated at each pixel (x, y) of Equation 5.2.6 can be made. For example, from iteration to iteration, the means \bar{u} and $\bar{v} = \overline{f - u}$,

and/or the standard deviations σ_u and $\sigma_v = \sigma_{f-u}$ can be taken to be constant. In fact, it was found in extensive experiments in which with (i) both the means and standard deviations constant, (ii) only the standard deviations constant, or (iii) neither the means or standard deviations constant, that there was very little difference in the obtained decomposition results. Therefore, for simplicity, both the means and standard deviations are kept constant from iteration to iteration, thus simplifying the Euler-Lagrange equations obtained from the energy term $E_{\text{decorrel}}(u)$ appearing in $E_{\text{DOSV}}(u)$.

If the means and standard deviations are taken to be constant from iteration to iteration, then σ_u and σ_v , the standard deviations of the cartoon and texture components respectively, can be taken out of the integral expression of $E_{\text{decorrel}}(u)$ so that

$$E_{\text{decorrel}}(u) = \frac{\gamma_d}{\sigma_u \sigma_v} \int_{\Omega} \left| E_{W(x,y)}(u - \bar{u})(f - u - \overline{(f - u)}) \right| dx dy. \quad (5.2.7)$$

Note here that $E_{W(x,y)}[\cdot]$ denotes the sample mean over the window $W(x,y)$ centered on (x,y) .

In fact, $\bar{v} = \overline{f - u}$ can be assumed to be equal to zero, as in [4], so that the expression becomes

$$E_{\text{decorrel}}(u) = \frac{\gamma_d}{\sigma_u \sigma_v} \int_{\Omega} \left| E_{W(x,y)}(u - \bar{u})(f - u) \right| dx dy \quad (5.2.8)$$

Although the property $\bar{v} = 0$ may not hold over all local windows $W(x,y)$ in the image, it was found experimentally that imposing this assumption did not significantly change the quality of obtained image decompositions, and since the result-

ing $\mathbf{E}_{\text{decorrel}}(u)$ in Equation 5.2.8 is simpler than that without this assumption, as in Equation 5.2.7, the former expression for $\mathbf{E}_{\text{decorrel}}(u)$ is taken.

5.2.3 Euler-Lagrange Equation

The first variation of $\mathbf{E}_{\text{decorrel}}(u)$ can be somewhat complicated to derive since in general \bar{u} is calculated from u values in the window $W(x, y)$, and may not be merely a constant. When the simplifying assumptions from the previous subsection are taken, so that the means and standard deviations of u and v are constant between one iteration and the next, and the mean of v is taken to be zero in any window $W(x, y)$, the first variation of $\mathbf{E}_{\text{decorrel}}(u)$ (from Equation 5.2.8) is found to be

$$\mathbf{E}'_{\text{decorrel}}(u) = \gamma_d \frac{E_{W(x,y)}[\text{sign}(\text{Cov}(u, f - u))](f - 2u + \bar{u})}{\sigma_u \sigma_{f-u}}. \quad (5.2.9)$$

In general, the gradient descent solution of the PDE(s) derived from minimization of an energy $\mathbf{E}(u)$ is obtained by solving the time-dependent PDE $u_t = -\mathbf{E}'(u)$, where $\mathbf{E}'(u)$ is the Euler-Lagrange equation for u , obtained from $\mathbf{E}(u)$ via the Calculus of Variations. In [4], Osher, Solé and Vese show that under some rather weak conditions, gradient descent can be replaced by the solution of the following PDE: $u_t = \Delta \mathbf{E}'(u)$, where Δ is the Laplacian operator. This is the approach taken in [4] to solve the OSV model, and since this model is now being extended, the same procedure is followed in this paper, though we have not verified the validity of the assumption that these conditions always hold for the proposed model. However, the obtained results indicate that this assumption still leads to better results than from the OSV model it is based upon. The PDE that must be solved for the

proposed decorrelated Osher-Solé-Vese (DOSV) model is

$$\begin{aligned} u_t &= \Delta E'_{\text{DOSV}}(u) \\ &= f - u - \frac{1}{2\lambda} \Delta \operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right) + \frac{1}{2\lambda} \Delta E'_{\text{decorrel}}(u) \end{aligned}$$

with $E'_{\text{decorrel}}(u)$ (without the Laplacian) given by Equation 5.2.9.

5.2.4 Test Images

The proposed DOSV decomposition model of this section was tested on a test set of four separate grayscale images, and the PMOSV and OLOSV decomposition models of the following section, were tested on a test set of three separate test images. The test set of images for DOSV consists of `smallbarbara` (a smaller version of another test image `barbara`), `lena`, `mandrill` and `grass`. The test image set for PMOSV/OLOSV is made up of `barbara`, `lena` and `grass`. The sizes of the images are as follows: `barbara`, `lena` and `mandrill` are 512x512 pixels large, `smallbarbara` is 256x256 pixels in size, and `grass` is 380 pixels wide and 332 pixels high. All these test images are shown in Figure 5.1.

Especially for PMOSV and OLOSV, such large images were used, since these depend on coherence of texture orientations, these coherences being higher when the texture is of slightly larger scale.



(a) barbara



(b) smallbarbara



(c) lena



(d) mandrill



(e) grass

Figure 5.1: Test Images used in this Chapter

5.2.5 Experimental Setup

The window $W(x, y)$, first appearing in Equation 5.2.4, and used in the simulations here, is taken to be 5 pixels by 5 pixels large. A larger window of 7x7 pixels was also tested, but did not lead to a significant improvement in decomposition results and required additional computations.

The value of γ_d , the coefficient of the new decorrelation term, and also first appearing in Equation 5.2.4, was set to 5.0. The fidelity parameter λ for both models was set to 0.015 for all iterations. Osher-Solé-Vese decomposition was run for 150 iterations on all test images, with explicit timestepping ($\Delta t = 0.0015$). A larger number (200) of iterations of DOSV decomposition was used on the test images, since it was found that in the initial stages of tests with that decomposition model, texture was extracted more slowly to the v component than by the OSV model. Once again, explicit timestepping was performed with $\Delta t = 0.0015$. For both OSV and DOSV decomposition, the same initial conditions are used, namely $u^0 = f$.

5.2.6 DOSV Experimental Results

5.2.6.1 Qualitative Results

As by Shahidi and Moloney in [68], and in the Improved Edge Segregation algorithm presented in Chapter 3 of this thesis, it is desired to reduce the presence of cartoon edges in the texture component of the outputs from image decomposition schemes. In Figure 5.2, the cartoon components u from the decomposition of `smallbarbara` with the OSV model and the just-described DOSV models from

Sections 5.2.1 and 5.2.3 are shown. Figure 5.3 shows the texture components v of the decomposition of the same image with the OSV and DOSV models. It is obvious that large scale features such as the edge of Barbara's arm and her facial features are much less evident in the texture component in Figure 5.3(b) than that of the conventional OSV model in Figure 5.3(a). Texture also appears to be slightly enhanced in this figure.

The DOSV algorithm was also tested on three other images, grass, mandrill and lena. For all three test images, there was a suppression of cartoon edges and enhancement of texture edges in the texture component v output from the proposed DOSV decomposition model as compared to the OSV decomposition model, DOSV is based upon. A qualitative comparison of the DOSV and OSV decompositions of the test image lena is made next.

For the test image lena, there was also a noticeable reduction of cartoon edges in the texture component v of DOSV decomposition as opposed to OSV decomposition. A zoom is shown in Figure 5.4. The cartoon edges associated with lena's hat are less prominent in the DOSV decomposition in Figure 5.4(b) than in the OSV decomposition in Figure 5.4(a), while the smaller scale texture edges in the feathers of the hat are stronger in the DOSV texture component than those in the OSV texture component.

The above comparisons between OSV and DOSV are qualitative in nature. Quantitative comparisons between the two decomposition algorithms are made next.



(a) Cartoon Component of OSV Decomposition of smallbarbara



(b) Cartoon Component of Proposed DOSV Decomposition of smallbarbara

Figure 5.2: Cartoon components of Decomposition of barbara with OSV and proposed DOSV models



(a) Texture Component of OSV Decomposition of *smallbarbara*



(b) Texture component of Proposed DOSV Decomposition of *smallbarbara*

Figure 5.3: Texture components of Decomposition of *smallbarbara* with OSV and proposed DOSV models



(a) Texture Component of OSV Decomposition of lena



(b) Texture component of Proposed DOSV Decomposition of lena

Figure 5.4: Texture components of Decomposition of lena with OSV and proposed DOSV models

5.2.6.2 Quantitative Results

To quantify the improvement in separation of cartoon and texture edges into their respective components, the decomposition quality measures of Section 3.7.5 are used, so that the qualitative comparisons in Section 5.2.6.1 are not relied upon exclusively.

Recall from Section 3.7.5, that there are two cartoon quality measures, CQ and CQ' , each measuring the average ratio of the cartoon edge strength $|\nabla u|$ to the original edge strength $|\nabla f|$ over different sets of cartoon edges, the set for CQ' a subset of that for CQ . Recall as well that in that section, there was defined a texture quality measure TQ measuring the average ratio of the texture edge strength $|\nabla v|$ to the original edge strength $|\nabla f|$. The higher any of these three measures is in its value, the better the decomposition is.

Test Image	Decomposition Method	CQ	CQ'	TQ
smallbarbara	OSV	0.9491	0.9623	0.7853
	DOSV	0.9688	0.9770	0.7997
lena	OSV	0.9527	0.9650	0.5551
	DOSV	0.9749	0.9769	0.5770
mandrill	OSV	0.9418	0.9440	0.7093
	DOSV	0.9794	0.9569	0.7357
grass	OSV	0.9566	0.9449	0.6927
	DOSV	0.9909	0.9625	0.7171

Table 5.1: Decomposition Quality at Cartoon and Texture Edges for OSV and proposed DOSV Decomposition Models

In Table 5.1, it is seen that all three decomposition quality measures are higher in value for all four test images DOSV was tested upon. Although the amount of improvement may not appear at first sight to be very substantial, it should be remembered that these quality measures are already quite high for the OSV decom-

position model, and so the amount of improvement is limited. However, despite the high decomposition quality measure values for the OSV model, low decomposition quality edges are still present. Significant improvements in edge quality for DOSV at certain pixels may result in a small overall increase in the decomposition quality measures, since an average is taken over a set of pixels, and these large improvements may be offset by less significant improvements at other pixels in the set. This fact is confirmed by the visual results in Section 5.2.6.1. Also observe that the values of the decomposition quality measures in Table 5.1 are higher in general than those for the Vese-Osher and IES models in Table 3.5. The fact that the OSV model outperforms the Vese-Osher model in better separating large-scale from small-scale edges in resulting decompositions has already been observed in [4].

In the following section, the next extension to the OSV model, based on the incorporation of nonlinear diffusion into the Euler-Lagrange equations of the OSV model, is introduced, for the purpose of denoising.

5.3 Incorporating Nonlinear Diffusion into the OSV Model

5.3.1 Perona-Malik Nonlinear Diffusion

Note that for the remainder of this chapter, decomposition is applied to the problem of textured image denoising. The noisy image f is split into the sum of a denoised image, corresponding to the cartoon component u , and the noise itself,

which is placed in the v component. As demonstrated in [4] by examples, the Osher-Solé-Vese decomposition (OSV) model is especially effective for denoising textured images in this way. In [78], it is shown that this is due to the higher weighting of lower frequencies in the texture component of an image using the H^{-1} -norm from the OSV model as opposed to the L^2 -norm, used e.g. in the Total Variation model of Equation 2.4.1. So high frequency and small scale noise is more likely to be placed in the texture component by decomposition with the OSV model compared to other models, e.g. the Rudin-Osher-Fatemi model and even the Vese-Osher model. In [4] and e.g. Section 5.2.1 of this thesis, it is shown that the energy functional for OSV decomposition can be minimized by evolving the cartoon component u according to the following partial differential equation

$$u_t = -\frac{1}{2\lambda}\Delta \left[\operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right) \right] - (u - f). \quad (5.3.1)$$

The second term of Equation 5.3.1 can be considered as a fidelity term, which keeps the cartoon component u close to the original noisy image f . However, it is proposed to introduce a nonlinear edge-stopping coefficient, as inspired by Perona and Malik's classic work [5], into the negative Laplacian of the curvature of u in the first term of the above evolution equation. Recall the discussion of this work in Section 2.2.1.3. This leads to the following PDE

$$u_t = -\frac{1}{2\lambda}\operatorname{div} \left(c(|\nabla u|)\nabla \left[\operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right) \right] \right) - (u - f). \quad (5.3.2)$$

Although it is not immediately obvious that this is the gradient descent solution of any energy functional easily expressible in closed form (and may in fact not be), the

use of the PDE can be justified by the following argument. The PDE used to solve the OSV decomposition model, given in Equation 5.3.1, is known experimentally to perform well when applied to the task of denoising. The second term in that equation $-(u - f)$ makes the cartoon component u gravitate towards the original image f , which may contain noise. Therefore, any denoising must come from the first term, proportional to (with positive ratio $\frac{1}{2\lambda}$) the negative Laplacian of the curvature of the level lines of u . Thus this denoising can be limited by including the edge-stopping coefficient $c(|\nabla u|)$ as in Equation 5.3.2.

The coefficient $c(|\nabla u|)$ is chosen as one of the two functions given in [5], namely

$$c(|\nabla u|) = \frac{1}{1 + \frac{|\nabla u|^2}{K^2}},$$

where K is a constant determined at each iteration based on the integral of gradient magnitudes in the image at that iteration. The value of K is set to be 0.9 times the integral of the gradient magnitudes over the entire image, as done in [5]. One thing to notice about the function c is that it is decreasing in its argument $|\nabla u|$. It is also important to emphasize the fact that in the finite difference discretization of $|\nabla u|$, the filters

$$D_x = \frac{1}{32} \begin{pmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{pmatrix} \quad (5.3.3)$$

and

$$D_y = \frac{1}{32} \begin{pmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{pmatrix} \quad (5.3.4)$$

were used to determine the partial derivatives of u with respect to x and y . These filters are more resistant to noise than the usual central difference scheme [31], so that noise will be diffused more easily by the proposed method. Noise resistance stems from the smoothing orthogonal to the magnitude of the gradient in the direction being measured. This new model is called by the rather lengthy concatenation of the inspiring authors of [5] and [4], the PMOSV decomposition model.

Note that the PMOSV model theoretically may not yield excellent results because the Perona-Malik diffusion flow is known to enhance edges [5], so that the “overshoot” of the edge caused by the enhancement may end up in the noise component v . However, it will be seen in Section 5.3.3 that PMOSV will inspire another decomposition model based on nonlinear diffusion, namely the Oriented-Laplacian OSV model, which does improve results over the OSV decomposition model.

5.3.2 Results for PMOSV

The original barbara image, which is 512x512 pixels large is shown in Figure 5.5(a). To this and the other test images, white Gaussian noise of standard deviation $\sigma = 20$ is added (see e.g. Figure 5.5(b)). Then both OSV decomposition and the proposed PMOSV model are run on the noisy image, until the SNR values hit their peak. A time step $\Delta t = 0.002$, is used for all images except for lena. For the lena test image, the denoising process became unstable for OSV decomposition, so that a smaller timestep of $\Delta t = 0.001$ was selected. For all test images, the parameter λ for OSV decomposition is determined by gradient projection as in [4] given the known noise standard deviation. The same is done for the proposed model except



(a) Original image barbara

(b) Image barbara with added noise

Figure 5.5: Original and noisy test image

that λ is kept constant for a small number of iterations (10) before gradient projection is used. This is done because the initial noisy image led to negative values of λ being calculated which in turn caused the noise to be amplified. The value of λ at each iteration for the PMOSV model (after the initial 10) is found using the following formula

$$\lambda = -\frac{1}{2\sigma^2} \int_{\Omega} c(|\nabla u|) \{ (f_x - u_x)K_x(u) + (f_y - u_y)K_y(u) \} dx dy, \quad (5.3.5)$$

where $K(u)$ is the curvature of u given by $K(u) = \text{div} \left(\frac{\nabla u}{|\nabla u|} \right)$.

As can be noticed, there is less texture present in the noise component of the proposed model (Figure 5.7(b)), than in the noise component of the OSV model (Figure 5.6(b)). This can be especially noticed in the stripe patterns of the pants and the bottom of the scarf. A quantitative comparison between OSV and PMOSV denoising is given in Table 5.2, where the obtained signal-to-noise ratios (SNRs) between the denoised images (corresponding to the u components) with OSV and



(a) Cartoon+Texture component u



(b) Noise component v

Figure 5.6: Denoising results from OSV Decomposition model



(a) Cartoon+Texture component u



(b) Noise component v

Figure 5.7: Denoising results from proposed PMOSV Decomposition model

Image	OSV SNR	PMOSV SNR
barbara	12.639	12.634
lena	16.528	14.666
grass	11.015	10.595

Table 5.2: SNRs for OSV denoising vs. PMOSV denoising on test images

PMOSV are compared. As usual, the SNR between the processed denoised image $I_{denoised}$ and the ideal denoised initial image I_0 is computed using the formula

$$SNR(I_0, I_{denoised}) = 10 \log_{10} \frac{\sum_{i=0}^M \sum_{j=0}^N (I_0(i, j) - \bar{I}_0)^2}{\sum_{i=0}^M \sum_{j=0}^N (I_{denoised}(i, j) - I_0(i, j))^2}, \quad (5.3.6)$$

where it is assumed that the two images are of size $M \times N$ pixels and \bar{I}_0 is the mean value of I_0 .

Contrary to what is expected from the qualitative results, the PMOSV SNRs are slightly worse than the OSV SNRs. This is because while the PMOSV model is better at keeping texture away from the noise component v (recall that there are only two components for these denoising experiments, with the v component corresponding to noise), it also keeps some noise, especially near large-scale edges. For example, this preservation of noise near large-scale edges can be seen on both sides of the table leg at the bottom-left of the barbara image.

The timings required for the two algorithms run using the same 2.6 GHz Pentium IV with 1 GB RAM as used in Chapters 2, 3 and 4 on the test images, are shown in Table 5.3. Once again implementation of the algorithms was done in MATLAB. Observe that these times are substantially greater than those found for previous algorithms, e.g. IES in Chapter 3 (see Table 3.4), in part because the test images themselves are much bigger (512x512 pixels).

Image	OSV		PMOSV	
	# of iterations	Time (s)	# of iterations	Time (s)
barbara	334	106.59	868	278.88
lena	1092	104.25	637	246.59
grass	196	28.78	457	74.28

Table 5.3: Time required for OSV denoising vs. PMOSV denoising on test images

Although the SNRs are not very high for PMOSV, it is believed that with some more work on this algorithm, results could be improved, e.g. by improving performance near cartoon edges.

5.3.3 Diffusion with Oriented Laplacians

As was seen in the previous section (Section 5.3.2), nonlinear Perona-Malik diffusion was found not to lead to a great improvement in decomposition quality over the plain OSV model. However, Weickert's coherence-enhancing diffusion as in [55] could be directly incorporated. The problem with this idea is that it is not suitable in a decomposition framework due to its closing of discontinuous structures and smoothing of corners. This enhancement of local coherence of structures in the image is largely due to the Gaussian blurring of the structure tensor from which the eigenvectors representing the diffusion axes are computed. An alternative to this which would be more in line with the general framework of decomposition is that of nonlinear diffusion with Oriented Laplacians [79]. In essence, this is very similar to the coherence enhancing diffusion of [55] without the Gaussian blurring of the structure tensor.

In the usual heat equation, $u_t = \Delta u = u_{xx} + u_{yy}$, there is diffusion equally in all directions. If at any point, a polar plot were made of how much diffusion occurs

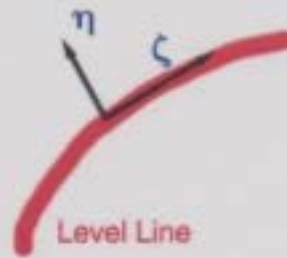


Figure 5.8: Illustration of Isophote and Gradient Directions

at every angle, with the radius $r(\theta)$ corresponding to the amount of diffusion at the angle θ , a circle would be obtained, because diffusion is isotropic (equal in all directions). Any two orthogonal directions ζ and η can be chosen, and an oriented Laplacian $\Delta_{\zeta\eta} = u_{\zeta\zeta} + u_{\eta\eta}$ defined. It can be easily shown that the Laplacian, corresponding to isotropic diffusion, is rotation invariant, so that in fact $\Delta_{\zeta\eta} = \Delta$, the standard Laplacian in Cartesian coordinates. Diffusion can be promoted along certain directions and inhibited in others, so that in general an ellipse is obtained for the polar diffusion plot. The advantage of the new representation of the Laplacian Δ , as an oriented Laplacian $\Delta_{\zeta\eta}$, is that coefficients can be added to each second directional derivative to limit the amount of diffusion in each of the orthogonal directions η and ζ , to form this elliptical polar diffusion plot.

So the general diffusion equation

$$u_t = c_\zeta u_{\zeta\zeta} + c_\eta u_{\eta\eta}, \quad (5.3.7)$$

can be obtained, as in [79].

If the vector ζ is chosen at every point to be the isophote direction, or the direction in which u changes the least, and η to be the gradient direction, or the direction in which u changes the most (see Figure 5.8), then smoothing can be pro-

moted along the isophotes by setting $c_\zeta = 1$ at all points. When there is a low gradient, there can be a large amount of smoothing in the gradient direction, but if there is a high gradient, no smoothing is desired across the edge in that direction. The spatially varying constant c_η can be set equal to $g(|\nabla u|)$, with $g(0) = 1$ and $g(x)$ approaching 0 as x becomes large. Then, isotropic diffusion will only be obtained in regions without a large gradient and only diffusion along isophotes at pixels around which a gradient is present.

However, the diffusion equation, Equation 5.3.7 only applies to image regions where there are well-defined gradient and isophote directions. The determination of such strongly oriented regions is now discussed.

5.3.4 Determining Oriented and Non-Oriented Regions

A region is defined to be oriented when its gradient direction coherence [55] is greater than a pre-determined threshold. This coherence function is a measure of how uniform the gradient directions are around a pixel.

To determine gradient direction at a pixel, the nonlinear structure tensor from Section 2.7.1 could be used. However this is rather inefficient because it requires an iterative process. Instead the linear structure tensor from the same section can be used, based on blurring of the following structure tensor J

$$J = \begin{pmatrix} f_x^2 & f_x f_y \\ f_x f_y & f_y^2 \end{pmatrix}.$$

Instead of using a Gaussian filter to blur J , an averaging filter of size 7x7 pixels

was used. Then the gradient direction $\theta_{i,j}$ at pixel (i, j) is determined to be the eigenvector corresponding to the larger eigenvalue of the average-filtered J at pixel (i, j) , and the isophote direction is determined to be the eigenvector corresponding to the smaller eigenvalue of this average-filtered J tensor.

In [56], coherence is measured directly using a small window W around each pixel by the formula

$$coher(\theta_{i,j}) = |\nabla g|_{i,j} \frac{\sum_{(u,v) \in W} ||\nabla g|_{u,v} \cos(\theta_{i,j} - \theta_{u,v})|}{\sum_{(u,v) \in W} |\nabla g|_{u,v}},$$

where $\theta_{i,j}$ is the gradient direction as determined above. The window W here is also chosen to be of size 7 pixels by 7 pixels. Because the above coherence measure can be calculated quite efficiently, it is used for the experiments in this chapter.

5.3.5 The Oriented-Laplacian Osher-Solé-Vese Image Decomposition Model

Recall the defining partial differential equation for OSV decomposition

$$u_t = -\frac{1}{2\lambda} \Delta K(u) - (u - f). \quad (5.3.8)$$

Instead of including the Laplacian of the curvature, $\Delta K(u)$, the oriented Laplacian of the curvature can be taken instead, by substituting the oriented Laplacian expression of the right-hand side of Equation 5.3.7 for the Laplacian in Equation

5.3.8. The new equation becomes

$$u_t = -\frac{1}{2\lambda}(c_\zeta K_{\zeta\zeta}(u) + c_\eta K_{\eta\eta}(u)) - (u - f). \quad (5.3.9)$$

Here ζ is the isophote direction of the curvature in the regions of the image where there is a dominant orientation, and η is the gradient direction of the curvature in such regions. Denote the union of the pixels in the image where there is a dominant orientation in the curvature $K(f)$ of the initial image f as $\Omega_O^{K(f)}$, and the rest of the image, where there is no such dominant orientation as $\Omega_{NO}^{K(f)}$. These regions are determined by thresholding the coherence, as measured in the previous subsection, of the curvature $K(f)$. In fact, the oriented regions could also be defined to be those where there is a dominant orientation in f itself, to obtain Ω_O^f and Ω_{NO}^f for the oriented and non-oriented regions respectively. This is the definition that is used, so that for the rest of this chapter, the superscripts are dropped and Ω_O is defined to be the set of pixels where the image f is oriented, and Ω_{NO} to be the set of pixels where it is not. This was done because the determination of oriented regions of curvature was substantially more sensitive to noise and discretization choices than that of oriented regions of the original image f . With the use of Ω_O^f and Ω_{NO}^f , there was found to be no decrease in the SNR of the overall denoised results (see Section 5.3.7). The threshold used for the coherence of the orientations of f is dependent on the image, and this parameter is called $coher_{thresh}$. In Ω_O , Equation 5.3.9 is used, whereas in Ω_{NO} , the ordinary evolution equation for Osher-Solé-Vese decomposition, as in Equation 5.3.8 is utilized. For the coefficient

c_η in Equation 5.3.9, the usual Perona-Malik diffusivity function

$$c_\eta = g(|\nabla K(u)|) = \frac{1}{1 + \frac{|\nabla K(u)|^2}{K_d^2}}, \quad (5.3.10)$$

could be chosen (with K_d set to 10). However, it was found that just setting c_η to zero gave similar results and was more simple, so the expression in Equation 5.3.10 was not used. Thus this coefficient is set to zero. Also c_ζ is set to a constant value of 1, so that there is full diffusion along the isophote direction. Substituting these values of the coefficients c_η and c_ζ into Equation 5.3.9 yields

$$u_t = -\frac{1}{2\lambda} K_{\zeta\zeta}(u) - (u - f), \quad (5.3.11)$$

for the iterative solution of the proposed decomposition model. The values of the second directional derivatives of the curvature in its isophote and gradient directions can be computed without actually calculating the isophote and gradient angles. This is done by using the general formula for the second-order directional derivative of the function $K(u)$ in the direction $\mathbf{w} = (w_x, w_y)$, i.e.

$$K_{\mathbf{w}\mathbf{w}} = w_x^2 K_{xx}(u) + 2w_x w_y K_{xy}(u) + w_y^2 K_{yy}(u),$$

and then substituting the unit gradient and isophote vector directions for \mathbf{w} . The unit gradient vector is simply

$$\mathbf{w}_{\text{grad}} = \left(\frac{K_x(u)}{\sqrt{K_x^2(u) + K_y^2(u)}}, \frac{K_y(u)}{\sqrt{K_x^2(u) + K_y^2(u)}} \right),$$

and the unit isophote vector is

$$\mathbf{w}_{\text{iso}} = \left(\frac{-K_y(u)}{\sqrt{K_x^2(u) + K_y^2(u)}}, \frac{K_x(u)}{\sqrt{K_x^2(u) + K_y^2(u)}} \right).$$

The formulae for the second directional derivative of the curvature in the isophote direction (ζ) and its gradient direction (η) are thus

$$K_{\zeta\zeta}(u) = \frac{-2K_x(u)K_{xy}(u)K_y(u) + K_{xx}(u)K_y^2(u) + K_x^2(u)K_{yy}(u)}{K_x^2(u) + K_y^2(u)} \quad (5.3.12)$$

$$K_{\eta\eta}(u) = \frac{K_x^2(u)K_{xx}(u) + 2K_x(u)K_{xy}(u)K_y(u) + K_y^2(u)K_{yy}(u)}{K_x^2(u) + K_y^2(u)}. \quad (5.3.13)$$

These expressions for the two directional derivatives can be substituted into Equation 5.3.9, with the function $K(u)$ representing the curvature of the cartoon component u . Better results were found to be obtained by using the normalized isophote orientation estimates on the original image f , $f_{\text{iso},x}$ and $f_{\text{iso},y}$, which are similar to the curvature isophote orientation, and then diffusing along the isophote direction. The quantities $f_{\text{iso},x}$ and $f_{\text{iso},y}$ were determined from the linear structure tensor (see Section 2.7.1). This is thought to be due to the sensitivity to noise of the formulae in Equations 5.3.12 and 5.3.13. So, the following approximation is obtained

$$K_{\zeta\zeta}(u) \approx f_{\text{iso},x}^2 K_{xx}(u) + 2f_{\text{iso},x}f_{\text{iso},y}K_{xy}(u) + f_{\text{iso},y}^2 K_{yy}(u). \quad (5.3.14)$$

The approximation in Equation 5.3.14 is subsequently substituted in the oriented Laplacian expression in Equation 5.3.11, to obtain the final evolution equation for

the proposed decomposition/denoising model

$$u_t = -\frac{1}{2\lambda}(f_{iso,x}^2 K_{xx}(u) + 2f_{iso,x}f_{iso,y}K_{xy}(u) + f_{iso,y}^2 K_{yy}(u)) - (u - f). \quad (5.3.15)$$

The proposed model is called the Oriented-Laplacian Osher-Solé-Vese (OLOS) decomposition model.

5.3.6 Implementation Details

Equation 5.3.15 is solved via explicit timestepping (time step $\Delta t = 0.002$) for all test images except for lena, which is more sensitive, and for which a smaller time step of 0.001 is needed. It was found that better denoising results were obtained for OLOS by using the first-order derivative coefficients $f_{iso,x}$ and $f_{iso,y}$ measured using the same filters as in PMOSV, which were

$$D_x = \frac{1}{32} \begin{pmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{pmatrix} \quad (5.3.16)$$

and

$$D_y = \frac{1}{32} \begin{pmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{pmatrix}. \quad (5.3.17)$$

As with other decomposition/denoising schemes, with known texture/noise variance, the value of λ is dynamically updated with iteration number using a

method based on gradient projection. In fact, in oriented regions (Ω_O) one dynamically updated value λ_O is used, and in non-oriented regions, a separate dynamically updated coefficient parameter λ_{NO} is calculated. Because there is no simple closed-form expression for the integrals with respect to x or y of $K_{\zeta\zeta}(u)$ or $K_{\eta\eta}(u)$, the expression obtained for λ_O is not simplified using integration by parts. The final formula that is obtained is

$$\lambda_O = \frac{1}{2\sigma^2} \int_{\Omega_O} (f - u)(c_\zeta K_{\zeta\zeta}(u) + c_\eta K_{\eta\eta}(u)) dx dy. \quad (5.3.18)$$

In non-oriented regions, integration by parts is also not used to find the value of λ_{NO} at each iteration. The expression for λ_{NO} at each iteration is

$$\lambda_{NO} = \frac{1}{2\sigma^2} \int_{\Omega_{NO}} (f - u) \Delta K(u) dx dy. \quad (5.3.19)$$

It was found that by not using integration by parts to compute λ , convergence was slower, but the entire denoising process was more stable (a larger step size could be used) and converged to a result denoised with a higher signal-to-noise ratio as compared to the ideal image uncorrupted by noise.

Results obtained for the three test images barbara, grass and lena are presented in the next subsection, from which it can be observed that, in general, denoised images with higher signal-to-noise ratios than those from OSV and PMOSV decomposition are obtained with OLOSV. In fact, it was found that the performance improvement of OLOSV over OSV decomposition was due in some part to the separate calculation of λ in both oriented and non-oriented regions of the test images, but that the oriented Laplacian calculation in OLOSV still gave substantial

Image	OSV SNR	PMOSV SNR	OSV2 SNR	OLOSV SNR
barbara	12.639	12.634	13.226	13.713
lena	16.528	14.666	16.887	16.994
grass	11.015	10.595	11.056	11.250

Table 5.4: SNRs for OSV denoising vs. PMOSV, OSV2 and OLOSV denoising on test images

Image	OSV		OSV2		OLOSV	
	Iters.	Time (s)	Iters.	Time (s)	Iters.	Time (s)
barbara	334	106.59	582	499.47 (+27.53)	630	718.89 (+30.30)
lena	1092	104.25	1523	1313.06 (+29.00)	1540	1734.84 (+29.42)
grass	196	28.78	215	94.63 (+11.61)	269	150.44 (+12.25)

Table 5.5: Time and Number of Iterations required for OSV denoising vs. OSV2 and OLOSV denoising on test images

improvement in the visual quality of the noise component v . In the next section, some examples are shown of OSV decomposition run with separate values of λ computed in the two types of image regions, and compared with decomposition results for OSV with a single image-wide λ value and OLOSV. The OSV algorithm with separate λ values in each region type is called OSV2.

5.3.7 Results for OLOSV

Table 5.5 shows the number of iterations and computation times required on a Pentium IV 2.6 GHz PC with 1 GB of RAM in a MATLAB environment, as in Chapters 3 and 4 for the OSV, OSV2 and OLOSV denoising algorithms. The PMOSV algorithm is omitted since the execution times for that algorithm were already given in Table 5.3. All four algorithms (OSV, PMOSV, OSV2 and OLOSV) were run once to determine at which iteration the SNR between the image being denoised and the original uncorrupted image reached its peak, and then run again for this number of

iterations. This was done to ensure fair comparison between the algorithms, since perhaps one algorithm would require fewer iterations to reach the peak SNR. In the OSV2 and OLOSV columns, each entry has a numerical value in brackets, referring to the extra amount of time needed to calculate the coherence of the curvature of the original image f , or f itself, using the linear structure tensor framework. This process was required in order to determine which part of the image was oriented (Ω_O), and which was non-oriented (Ω_{NO}). Observe that in Table 5.5, even OSV2 with the separate calculation of λ_O and λ_{NO} in oriented and non-oriented regions, leads to a substantial increase in running time over the plain OSV algorithm (e.g. 1313.06 seconds for OSV2 on grass vs. 104.25 seconds for OSV on the same image). Though some of this difference in running time is due to the difference in the number of iterations required to reach the SNR peak for both algorithms, the actual calculation of the λ values for each iteration in oriented and non-oriented regions also leads to an increase in running time for OSV2. Additionally, this separate calculation of λ_O and λ_{NO} is thus also a major factor in the slow running time of the proposed OLOSV algorithm. This slowdown could perhaps be mitigated by only dynamically calculating these λ values every several iterations, instead of every iteration.

The confidence threshold $coher_{thresh}$ for the coherence depended on the image being denoised, and was equal to 20 for barbara and lena, and 25 for grass. These confidence threshold values were determined experimentally, so that the determined oriented regions corresponded qualitatively to be roughly the same as those which would be determined visually from the test images, though the overall results were not extremely sensitive to the thresholds chosen. As long as actual non-

oriented pixels were not considered to belong to the oriented part of the image Ω_O , in which case Equation 5.3.15 would be applied to these non-oriented pixels, the qualitative and quantitative denoising results obtained from OLOSV were better than those from OSV or OSV2.

Table 5.4 gives the signal-to-noise ratios of the denoised results from OSV decomposition, and from the extensions combining it with the Perona-Malik and Oriented-Laplacian nonlinear diffusion models, along with OSV2 decomposition. Recall that OSV2 decomposition is the same as OSV decomposition except that the λ parameter is calculated separately in oriented and non-oriented regions. As can be seen, the Oriented-Laplacian SNRs are the best, but of course this is at the expense of extra computation time, as shown in Table 5.5.

From Table 5.4, it is observed that some of the improvement associated with OLOSV is due to the separate computation of the λ coefficient in oriented and non-oriented regions, since there is a substantial SNR gain of OSV2 over OSV. But there is another substantial gain of OLOSV over OSV2, and OLOSV still reduces the amount of oriented texture in the v component from OSV2, and keeps the texture in the denoised cartoon component u .

Denoising results from the test image barbara are shown in Figures 5.9 (denoised cartoon components u) and 5.10 (noise components v). The texture for example on the pants of barbara and on her scarf are much sharper and more prominent in the OLOSV result of Figure 5.9(c) than in the OSV result of Figure 5.9(a) and the OSV2 cartoon component of Figure 5.9(b). Also, for the noise components in Figure 5.10, there is much less oriented texture in the OLOSV noise component v of Figure 5.10(c) than in the v components of Figures 5.6(b) and 5.10(b) for OSV and OSV2



(a) OSV Cartoon component of barbara



(b) OSV2 Cartoon component of barbara



(c) OLOS Cartoon component of barbara

Figure 5.9: Cartoon (denoised) components from OSV, OSV2 and OLOS Decompositions of barbara image



(a) OSV Noise component of barbara



(b) OSV2 Noise component of barbara



(c) OLOS Noise component of barbara

Figure 5.10: Noise components from OSV, OSV2 and OLOS Decompositions of barbara image



(a) Zoom on Noise component v of barbara from OSV model
 (b) Zoom on Noise component v of barbara from OSV2 model
 (c) Zoom on Noise component v of barbara from OLOS model

Figure 5.11: Zoom on Noise Components from OSV, OSV2 and OLOS decompositions of barbara image

respectively. A zoom of the same small portion of this component for the barbara image is shown in Figure 5.11, where this difference is more obvious.

The difference in the amount of oriented texture within the u and v components in Figures 5.12 and 5.13 respectively, between the three algorithms for the test image grass is less visible, but can be seen more clearly in the zoom of the v components in Figure 5.14.

5.4 Conclusions

This chapter introduced two proposed extensions to the OSV model of Section 2.4.3. The Decorrelated OSV (DOSV) model from Section 5.2 added a term which kept samples of the local correlation field small in value. Good results similar to those from IES in Chapter 3 were obtained, with cartoon and texture information better separated into their respective decomposition components.



(a) OSV Cartoon component of grass



(b) OSV2 Cartoon component of grass



(c) OLOSV Cartoon component of grass

Figure 5.12: Cartoon (denoised) components from OSV, OSV2 and OLOSV Decompositions of grass image



(a) OSV Noise component of grass

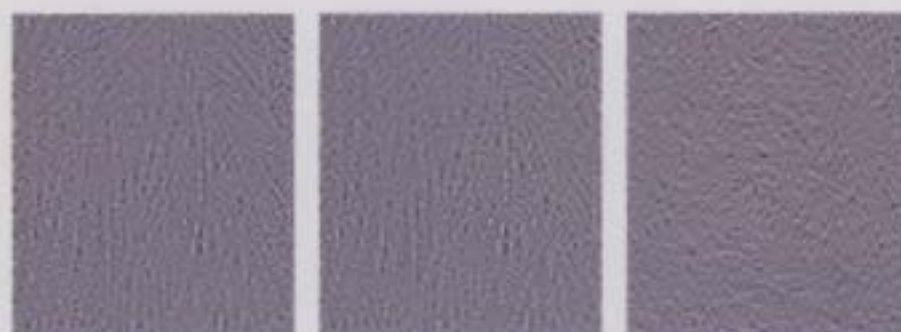


(b) OSV2 Noise component of grass



(c) OLOS Noise component of grass

Figure 5.13: Noise components resulting from OSV, OSV2 and OLOS Decompositions of grass image



(a) Zoom on Noise component v of grass from OSV model (b) Zoom on Noise component v of grass from OSV2 model (c) Zoom on Noise component v of grass from OLOSV model

Figure 5.14: Zoom on Noise Components from OSV, OSV2 and OLOSV Decompositions of grass image

Then two decomposition models combining the decomposition and nonlinear diffusion frameworks, for the purposes of denoising, were developed. The Oriented-Laplacian OSV (OLOSV) model of Section 5.3.5 was especially effective for the denoising of oriented texture. Additionally, it was found from the experiments in this chapter that cartoon edges were not as visible in the noise components v of the OLOSV results when compared with the OSV noise components. This is because a directional diffusion was performed close to these cartoon edges if they were calculated as being in the coherent regions of the image, and along the direction of the edges, not perpendicular to them. Unfortunately, there was slight visible smearing of the noise close to the edges, but this could be reduced by only including pixels on cartoon edges in the region in the image computed to have a coherent orientation, and not those around. The detection of such cartoon edges could be implemented with a Total-Variation-like diffusion, as done in Chapter 3.

It may be argued that the Oriented Laplacian OSV model used for denoising in this section is very similar to ordinary Oriented Laplacian diffusion [79]. How-

ever, the OLOSV model has two main differences from plain Oriented Laplacian diffusion:

1. Mathematically it is a novel nonlinear diffusion flow based on the Laplacian of the level set curvature of the cartoon component, instead of the Laplacian of the image, and
2. It is based on a model for image decomposition, and thus variations of it could potentially be used for the other applications of image decomposition, e.g. inpainting.

As for the second point above, the OLOSV model as presented in this chapter cannot be directly applied to pure decomposition, but could perhaps be altered to do so. In summary, the OLOSV model is a very powerful one for the denoising of oriented texture, and offers substantial performance improvements over the OSV model on which it is based. This problem of denoising oriented texture has important applications to medical imaging, for example for Diffusion-Tensor Magnetic Resonance Imaging (DT-MRI) [80] and to forensics, for example as preprocessing for the recognition of fingerprints [81].

Orientation-Adaptive Decomposition

6.1 Introduction

In Chapter 3 and in the papers by Shahidi and Moloney [82, 68], the Improved Edge Segregation image decomposition model was introduced, and it was shown that it gave better decomposition results than the standard Vese-Osher method and also led to improved texture discrimination results. In this chapter, a new model called Orientation-Adaptive Image Decomposition (initially presented in [83, 84]) is introduced which models the texture component v of an oriented image with just one subcomponent instead of the two required by virtually all existing models, e.g. [1, 68], with the exception of the Osher-Solé-Vese model [4]. Although the texture component is only comprised of one *sub*component, the proposed model could be strictly considered to be a three-component model - the u and v components are as defined from [2], but the r component is also important because it is in this component that the noise in the image is placed. This is similar to very mathematically complex three-component models, such as that of Aujol [39], but

in those models, the noise is explicitly modelled and placed in a third component w , and there is a fourth residual term. The proposed model is significant because it is efficient, simple to solve, gives good decomposition results, and provides a novel method for the denoising of oriented texture.

6.1.1 Motivation for the Simplification of the Vese-Osher Decomposition Model for Oriented Images

As already given in several previous chapters in this thesis, the Vese-Osher energy functional, the first practical method to minimize Meyer's variational decomposition model [1], is based upon the following energy functional

$$\mathbf{E}_{\mathbf{VO}}(u, g_1, g_2) = \int_{\Omega} |\nabla u| dx dy + \lambda \int_{\Omega} (f - u - g_{1,x} - g_{2,y})^2 dx dy + \mu \int_{\Omega} \sqrt{g_1^2 + g_2^2} dx dy. \quad (6.1.1)$$

The first term is a total variation term, ensuring that u is piecewise smooth, the second term is a fidelity term and the third models v in terms of the subcomponents g_1 and g_2 as texture. The Euler-Lagrange equations for the above functional can be easily calculated [2]. The three equations in each of u , g_1 and g_2 are

$$K(u) = 2\lambda(u + g_{1,x} + g_{2,y} - f) \quad (6.1.2)$$

$$\mu \frac{g_1}{\sqrt{g_1^2 + g_2^2}} = 2\lambda(u_x + g_{1,xx} + g_{2,xy} - f_x) \quad (6.1.3)$$

$$\mu \frac{g_2}{\sqrt{g_1^2 + g_2^2}} = 2\lambda(u_y + g_{1,xy} + g_{2,yy} - f_y) \quad (6.1.4)$$

where $K(u) = \text{div}\left(\frac{\nabla u}{|\nabla u|}\right)$ is the curvature of the level lines of u . Observe that the right-hand side of Equation 6.1.3 is equal to the partial derivative with respect to x of this curvature (from Equation 6.1.2), and that similarly, the right-hand side of Equation 6.1.4 is equal to the partial derivative with respect to y of the same quantity. Therefore, Equations 6.1.3 and 6.1.4 become

$$\mu \frac{g_1}{\sqrt{g_1^2 + g_2^2}} = K_x(u) \quad (6.1.5)$$

$$\mu \frac{g_2}{\sqrt{g_1^2 + g_2^2}} = K_y(u). \quad (6.1.6)$$

Also observe that the fidelity term in the V-O functional (Equation 6.1.1) which imposes the condition that $f - u \approx g_{1,x} + g_{2,y}$, includes a partial derivative with respect to x of g_1 , and a partial derivative with respect to the same variable is taken of the curvature in Equation 6.1.5 relating g_1 and $K(u)$. A similar relation holds for g_2 .

This suggests varying the direction of the partial derivatives so that one of the derivatives consistently vanishes, also called taking gauge coordinates. This is shown in Figure 6.1 where the coordinates at a point are taken in the gradient and isophote directions. This idea is now developed in the following section.

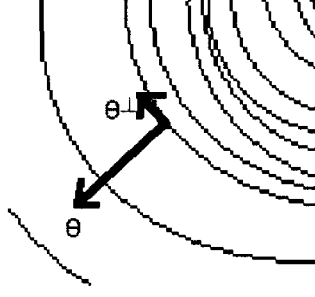


Figure 6.1: Example of isophote and gradient directions (θ^\perp and θ respectively)

6.2 Orientation-Adaptive Decomposition Model

6.2.1 Derivation

In Meyer's work [1], the norm of a function v on the Banach space G is defined as

$$\|v\|_* = \|\sqrt{g_1^2 + g_2^2}\|_{L^\infty}, \quad (6.2.1)$$

over all bounded g_1, g_2 such that $v = g_{1,x} + g_{2,y}$. A local rotation $\theta(x, y)$ is introduced in an effort to eliminate g_2 . Define the gradient direction of $K(u)$ to be $\theta(x, y)$, and the isophote direction to be $\theta^\perp(x, y) = \theta(x, y) + \frac{\pi}{2}$. So now generalize the model of Meyer, and suppose that the norm of v is the same as in Equation 6.2.1 except that it is taken as the minimum over all bounded g_1 and g_2 such that

$$v = \partial_\theta g_1 + \partial_{\theta^\perp} g_2 = \cos \theta g_{1,x} + \sin \theta g_{1,y} - \sin \theta g_{2,x} + \cos \theta g_{2,y}. \quad (6.2.2)$$

The original Meyer model [1] with the $*$ -norm can be recovered by setting $\theta(x, y) = 0$. Now it is attempted to eliminate g_2 by an appropriate choice of $\theta(x, y)$ with constrained minimization of an appropriate energy functional.

Define the energy functional

$$\begin{aligned} \mathbf{E}_{\text{ROT}}(u, g_1, g_2, \theta) = & \int_{\Omega} |\nabla u| dx dy + \mu \int_{\Omega} \sqrt{g_1^2 + g_2^2} dx dy + \\ & \lambda \int_{\Omega} (f - u - \partial_{\theta} g_1 - \partial_{\theta^{\perp}} g_2)^2 dx dy, \end{aligned}$$

where $\partial_{\theta} g_1 + \partial_{\theta^{\perp}} g_2$ is as defined in Equation 6.2.2. Using the calculus of variations, the following Euler-Lagrange equations for u , g_1 and g_2 are obtained

$$K(u) = -2\lambda(f - u - \partial_{\theta} g_1 - \partial_{\theta^{\perp}} g_2) \quad (6.2.3)$$

$$\mu \frac{g_1}{\sqrt{g_1^2 + g_2^2}} = \frac{d}{dx} [2\lambda(f - u - \partial_{\theta} g_1 - \partial_{\theta^{\perp}} g_2)(-\cos \theta)] + \quad (6.2.4)$$

$$\begin{aligned} & \frac{d}{dy} [2\lambda(f - u - \partial_{\theta} g_1 - \partial_{\theta^{\perp}} g_2)(-\sin \theta)] \\ \mu \frac{g_2}{\sqrt{g_1^2 + g_2^2}} = & \frac{d}{dx} [2\lambda(f - u - \partial_{\theta} g_1 - \partial_{\theta^{\perp}} g_2)(\sin \theta)] + \quad (6.2.5) \\ & \frac{d}{dy} [2\lambda(f - u - \partial_{\theta} g_1 - \partial_{\theta^{\perp}} g_2)(\cos \theta)]. \end{aligned}$$

The first equation can now be substituted into the second and third to get

$$\mu \frac{g_1}{\sqrt{g_1^2 + g_2^2}} = (K(u) \cos \theta)_x + (K(u) \sin \theta)_y \quad (6.2.6)$$

$$\mu \frac{g_2}{\sqrt{g_1^2 + g_2^2}} = (-K(u) \sin \theta)_x + (K(u) \cos \theta)_y. \quad (6.2.7)$$

Assuming that locally $\theta(x, y)$ is almost constant, the $\sin \theta$ and $\cos \theta$ can be taken

out of the partial derivatives in Equation 6.2.7 to yield

$$\begin{aligned}\mu \frac{g_2}{\sqrt{g_1^2 + g_2^2}} &= -K_x(u) \sin \theta + K_y(u) \cos \theta \\ &= K(u)_{\theta^\perp} \approx 0 \Rightarrow g_2 \approx 0.\end{aligned}$$

since θ^\perp was defined to be the isophote direction of the curvature $K(u)$. Thus, it has been shown that with an appropriate choice of the local rotation angle, $\theta(x, y)$, the texture subcomponent g_2 can be made negligible. Because only g_1 remains, it is renamed g .

Next, with the assumption that $g_2 = 0$, a new energy $E_{\text{OAD}}(u, g, \theta)$ is defined

$$\begin{aligned}E_{\text{OAD}}(u, g, \theta) &= \int_{\Omega} |\nabla u| dx dy + \mu \int_{\Omega} |g| dx dy + \\ &\quad \lambda \int_{\Omega} (f - u - \cos \theta g_x - \sin \theta g_y)^2 dx dy,\end{aligned}\tag{6.2.8}$$

The third term is the new soft constraint fidelity term

$$f - u \approx \cos \theta g_x + \sin \theta g_y \tag{6.2.9}$$

and the second term is now so defined because $g_2 = 0$.

The angular function θ has already been constrained to be the gradient direction of the curvature of the level lines of the cartoon component, i.e. $\theta = \arctan \left(\frac{K_y(u)}{K_x(u)} \right)$. With the assumption that $\theta(x, y)$ doesn't change too much locally, the gradient angle of $K(u)$ can be approximated by that of the original f , assuming that the latter is measured in a noise-resistant manner. Recall that a similar approximation was

taken in Chapter 5 for the OLOS algorithm. Thus, θ can be calculated once, before the Euler-Lagrange iterations, thereby effectively reducing the texture information that needs to be dynamically updated to the one subcomponent g .

The gradient direction measurement of f , as described in the next section, is relatively insensitive to noise. Thus, the proposed one-subcomponent model can be shown to be an effective model for both decomposition and denoising. It was found empirically that since the noise typically doesn't have a preferred orientation, it is mostly transferred to the residual. Also, by setting g_2 to be 0, the implicit constraint is being added that the directional derivative of the curvature $K(u)$ in its isophote direction is zero, thus eliminating any noise or other irregularities along this isophote. The proposed model, based on the energy in Equation 6.2.9, is called Orientation-Adaptive Decomposition (OAD).

6.2.2 Solution of Proposed Model

The Euler-Lagrange equations for the proposed one-subcomponent model with energy from Equation 6.2.9 are

$$\operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right) + 2\lambda(f - u - g_\theta) = 0 \quad (6.2.10)$$

$$\begin{aligned} \mu \operatorname{sign}(g) &= 2\lambda(u_\theta + g_{\theta\theta} - f_\theta) + 2\lambda(f - u - g_\theta) ((-\cos \theta)_x + (-\sin \theta)_y) \\ &\Rightarrow \mu \operatorname{sign}(g) \approx 2\lambda(u_\theta + g_{\theta\theta} - f_\theta). \end{aligned} \quad (6.2.11)$$

The last equation includes the assumption that θ does not change too much locally, so that the second term in the Euler-Lagrange equation for g vanishes. The

equations are solved by a semi-implicit fixed-point scheme as in [2], where the curvature of u term is kept in divergence form. The discretization of the various directional derivatives in the above equations is discussed below in Section 6.3.1. The initial conditions $u^0 = f$ and $g^0 = 0$ are chosen, along with Neumann boundary conditions for u and Dirichlet boundary conditions for g . Recall that in practice, Neumann boundary conditions refer to padding the function with repetition around the border of its domain, and that Dirichlet boundary conditions refer to zero-padding.

It is necessary to compute the orientation field of the original image f in order to obtain the angles $\theta(i, j)$ at each pixel. As described in Section 2.7.1, though the nonlinear structure tensor [54] could be used to compute the orientation field of the image f very accurately, the linear structure tensor is preferred due to its increased efficiency. Thus, the linear structure tensor is used in the experiments in this chapter.

6.3 Division of Orientations into Sectors

It was found empirically by the experiments in this thesis that the given OAD model works gives the best decomposition results for gradient angles θ such that $\tan(\theta)$ is close to -1. In fact, the model was found to work extremely well for angles θ in the set

$$P_\theta = \left\{ \theta \text{ such that } \left| \theta - \frac{3\pi}{4} \right| \leq \frac{\pi}{8} \text{ or } \left| \theta + \frac{\pi}{4} \right| \leq \frac{\pi}{8} \right\}. \quad (6.3.1)$$

In this case, the angles θ are considered modulo 2π , and restricted to the usual 2π interval $(-\pi, \pi]$. Similar angle preference in decomposition output is observed with Vese-Osher decomposition. However after several iterations of their method this effect was found to dissipate.

6.3.1 Rotation of Derivative Filters

In the experiments in this chapter, OAD was applied to both decomposition by itself, and to decomposition with denoising, both for images with oriented texture. It was found in initial experiments that the OAD algorithm is very sensitive to the choice of derivative filters used. It is especially important that the derivative filters used should be as insensitive as possible to rotation, in particular for decomposition, and less so for denoising. In Sections 6.3.1.1 and 6.3.1.2, the choices of these derivative filters are discussed, for the application of OAD to pure decomposition and to denoising respectively.

6.3.1.1 Rotated Derivative Filters for Decomposition

Farid and Simoncelli [85] used an optimization procedure to find the derivative filters in the x and y directions of various small extents with the least integrated error when used to find directional derivative estimates across all angles θ . For first-order directional derivatives, for example, the standard formula $D_\theta = \cos(\theta)D_x + \sin(\theta)D_y$ is used to find the directional derivatives, where D_x and D_y are the derivative filters in the x and y directions respectively. It was found in this research that the best choices of these derivative filters D_x and D_y for decomposition

were the 3-tap filters from [85]. The 5-tap filters from [85] were found to be the best choices for the second derivative filters, D_{xx} , D_{xy} and D_{yy} for decomposition.

In [85], the derivative filters are assumed to be separable, and are found by differentiating an interpolation filter. This amounts to using a smoothing pre-filter p in the direction perpendicular to the derivative filter d . The actual numerical values of the first-order filters are found with the formulae

$$\begin{aligned} D_x(x, y) &= d_1(x)p_1(y) \\ D_y(x, y) &= p_1(x)d_1(y), \end{aligned} \tag{6.3.2}$$

where

$$\begin{aligned} d_1(x) &= \begin{bmatrix} -0.42529 & 0 & 0.42529 \end{bmatrix}^T \text{ and} \\ p_1(y) &= \begin{bmatrix} 0.22988 & 0.54024 & 0.22988 \end{bmatrix}, \end{aligned}$$

and $d_1(y) = d_1^T(x)$ and $p_1(x) = p_1^T(y)$, with the T superscript denoting the transpose operator. For the second-order derivatives, the expressions

$$\begin{aligned} D_{xx}(x, y) &= e_2(x)p_2(y) \\ D_{xy}(x, y) &= e_1(x)e_1(y) \\ \text{and } D_{yy}(x, y) &= p_2(x)e_2(y) \end{aligned} \tag{6.3.3}$$

are used, where

$$\begin{aligned} e_2(x) &= \begin{bmatrix} 0.23291 & 0.00267 & -0.47115 & 0.00267 & 0.23291 \end{bmatrix}^T, \\ p_2(y) &= \begin{bmatrix} 0.03032 & 0.24972 & 0.42638 & 0.24972 & 0.03032 \end{bmatrix} \\ e_1(x) &= \begin{bmatrix} -0.10455 & -0.29232 & 0 & 0.29232 & 0.10455 \end{bmatrix}^T. \end{aligned}$$

Similar to the first-derivative filters, $e_1(y) = e_1^T(x)$, $e_2(y) = e_2^T(x)$ and $p_2(x) = p_2^T(y)$. All these equations are expressed in terms of the usual MATLAB image coordinates.

As described in the previous section, these derivative filters are most effective for angles in the set P_θ (as defined in Equation 6.3.1). For other orientations, the derivative filters have to be rotated. The only rotations which are needed are by angles $\pm \frac{\pi}{4}$ and $+\frac{\pi}{2}$. Let

$$P_1 = P_{\theta - \frac{\pi}{4}} = \left\{ \theta \text{ such that } \theta - \frac{\pi}{4} \in P_\theta \right\} \quad (6.3.4)$$

$$P_2 = P_{\theta + \frac{\pi}{4}} = \left\{ \theta \text{ such that } \theta + \frac{\pi}{4} \in P_\theta \right\} \quad (6.3.5)$$

$$P_3 = P_{\theta + \frac{\pi}{2}} = \left\{ \theta \text{ such that } \theta + \frac{\pi}{2} \in P_\theta \right\} \quad (6.3.6)$$

For simplicity and consistency, let P_θ be P_0 . The sectors $\{P_i\}_{i=0}^3$ are shown diagrammatically in Figure 6.2. When the angle θ lies in each of the angular sectors P_1 , P_2 and P_3 as shown in Figure 6.2, the corresponding derivative filters are defined by Tables 6.1 and 6.2 in terms of the filters D_x , D_y , D_{xx} , D_{xy} and D_{yy} for angular region P_0 .

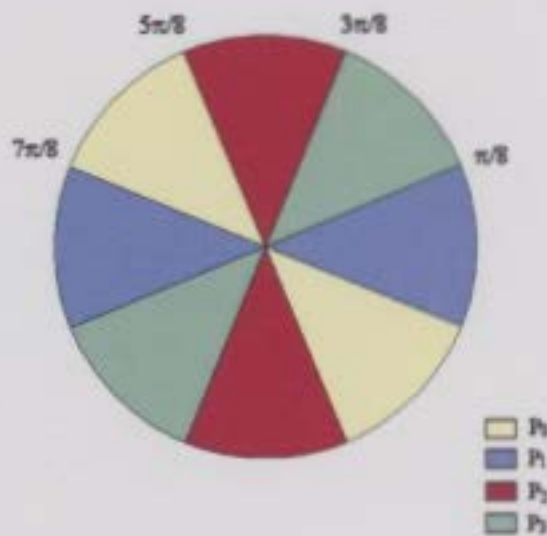


Figure 6.2: Regions P_i with different derivative filters for decomposition (not de-noising)

Domain	First order	
	x	y
P_0	D_x	D_y
P_1	$\frac{1}{\sqrt{2}}(D_x + D_y)$	$\frac{1}{\sqrt{2}}(D_y - D_x)$
P_2	$\frac{1}{\sqrt{2}}(D_x - D_y)$	$\frac{1}{\sqrt{2}}(D_x + D_y)$
P_3	$-D_y$	D_x

Table 6.1: First-Order Derivative Filters for each Angular Sector $\{P_i\}_{i=0}^3$

Domain	Second order		
	xx	xy	yy
P_0	D_{xx}	D_{xy}	D_{yy}
P_1	$\frac{1}{2}(D_{xx} + 2D_{xy} + D_{yy})$	$\frac{1}{2}(D_{yy} - D_{xx})$	$\frac{1}{2}(D_{xx} - 2D_{xy} + D_{yy})$
P_2	$\frac{1}{2}(D_{xx} - 2D_{xy} + D_{yy})$	$\frac{1}{2}(D_{xx} - D_{yy})$	$\frac{1}{2}(D_{xx} + 2D_{xy} + D_{yy})$
P_3	D_{yy}	$-D_{xy}$	D_{xx}

Table 6.2: Second-Order Derivative Filters for each Angular Sector $\{P_i\}_{i=0}^3$

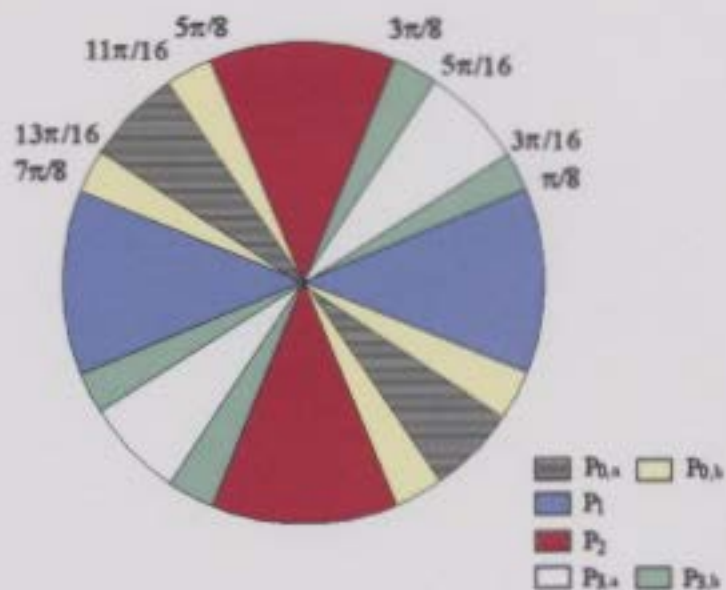


Figure 6.3: Regions P_i with different derivative filters for denoising

6.3.1.2 Rotated Derivative Filters for Denoising

Although the use of these derivative filters was very effective for the purposes of image decomposition, on the other hand for denoising, simpler first and second-order derivative filters of small size gave better results. The region where OAD was found to work well for denoising was slightly smaller than that for pure decomposition

$$P_{0,a} = \left\{ \theta \text{ such that } \left| \theta - \frac{3\pi}{4} \right| \leq \frac{\pi}{16} \text{ or } \left| \theta + \frac{\pi}{4} \right| \leq \frac{\pi}{16} \right\}. \quad (6.3.7)$$

This leads to angular regions for denoising slightly different than those for decomposition which were shown in Figure 6.2. Figure 6.3 shows these regions for the task of denoising. The regions $P_{0,b}$, $P_{3,a}$ and $P_{3,b}$ are defined later in this section, but can be seen in Figure 6.3. The first-order derivative filters D_x and D_y in P_0 both

had size 3x3 pixels, and are based on central differences. In P_0 , these first-order filters are

$$D_x = \frac{1}{2} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}, D_y = \frac{1}{2} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}. \quad (6.3.8)$$

For sector $P_{0,a}$, the second-order derivative filters are also based on central differences, and are given by

$$D_{xx} = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}, D_{xy} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}, D_{yy} = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}. \quad (6.3.9)$$

The region $P_{3,a}$ is defined to be the same as $P_{0,a}$ except flipped along the y axis

$$P_{3,a} = \left\{ \theta \text{ such that } \left| \theta - \frac{\pi}{4} \right| \leq \frac{\pi}{16} \text{ or } \left| \theta + \frac{3\pi}{4} \right| \leq \frac{\pi}{16} \right\}. \quad (6.3.10)$$

In $P_{3,a}$,

$$D_x = \frac{1}{2} \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \text{ and } D_y = \frac{1}{2} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \quad (6.3.11)$$

The second-order derivative filters for $P_{3,a}$ can be derived from those in Equation 6.3.9: D_{xx} and D_{yy} are switched, and \hat{D}_{xy} for $P_{3,a}$ is obtained from D_{xy} for $P_{0,a}$ by

flipping the filter matrix horizontally to obtain

$$\hat{D}_{xy} = \begin{bmatrix} 0 & -\frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & 0 \end{bmatrix}.$$

The regions $P_{0,b}$ and $P_{3,b}$ are defined to satisfy $P_0 = P_{0,a} \cup P_{0,b}$ and $P_3 = P_{3,a} \cup P_{3,b}$, where P_0 and P_3 are as defined in Section 6.3.1.1. The first-order derivative filters for $P_{0,b}$ are the same as those for $P_{0,a}$, and those for $P_{3,b}$ are the same as those for $P_{3,a}$ in Equation 6.3.11. The second-order derivative filters are however defined differently; for $P_{0,b}$ and $P_{3,b}$,

$$D_{xx} = D_x * D_x, D_{xy} = D_x * D_y, D_{yy} = D_y * D_y, \quad (6.3.12)$$

where $\{D_x, D_y\}$ is the set of first-order derivative filters for $P_{0,b}$ and $P_{3,b}$ respectively.

For P_1 , the first-order derivative filters are

$$D_x = \frac{1}{32} \left(\frac{1}{\sqrt{2}} \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} + \frac{1}{\sqrt{2}} \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} \right), \quad (6.3.13)$$

$$D_y = \frac{1}{32} \left(-\frac{1}{\sqrt{2}} \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} + \frac{1}{\sqrt{2}} \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} \right). \quad (6.3.14)$$

For P_2 , these first-order derivative filters are

$$D_x = \frac{1}{32} \left(\frac{1}{\sqrt{2}} \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} - \frac{1}{\sqrt{2}} \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} \right), \quad (6.3.15)$$

$$D_y = \frac{1}{32} \left(\frac{1}{\sqrt{2}} \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} + \frac{1}{\sqrt{2}} \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} \right). \quad (6.3.16)$$

Finally, for orientation sectors P_1 and P_2 , the second-order derivative filters are obtained from those of Farid and Simoncelli (D_{xx} , D_{xy} and D_{yy}), as found in Equation 6.3.4. The second-order derivative filters in sector P_1 , \tilde{D}_{xx} , \tilde{D}_{xy} and \tilde{D}_{yy} , are almost the same as those that were used for pure decomposition (except for the use of D_{xy}^H instead of $-D_{xy}$), and are given by

$$\tilde{D}_{xx} = \frac{1}{2}(D_{xx} + 2D_{xy}^H + D_{yy}), \quad \tilde{D}_{xy} = -\frac{1}{2}(D_{yy} - D_{xx}), \quad \tilde{D}_{yy} = \frac{1}{2}(D_{xx} + 2D_{xy} + D_{yy}), \quad (6.3.17)$$

where the H superscript denotes a horizontal flip of the matrix.

6.4 Boundary Conditions

For the entire image, the cartoon component u is padded adiabatically (with repetition), and the texture subcomponent g is padded linearly out from each image boundary, so that v will roughly be padded adiabatically as well. Since g has different meanings in oriented and non-oriented regions, a similar padding was used

for g between these regions. However, because the inter-region boundaries are in general irregular, each region's boundary condition consists of a weighted sum of g values from the same type of region (oriented or non-oriented) with weights inversely related to distance from the boundary. Also the oriented regions are morphologically closed with a disk element of radius 3, to make the boundary more regular, because an irregular boundary results in the weighted sum being taken from pixels all around the boundary pixel, and because these could have very different values, instability could result.

Even within the oriented part of the image, there are many orientation sectors P_i as explained in Section 6.3. Boundary conditions are necessary for these orientation sectors as well, as explained in the next section.

6.4.1 Boundary Conditions between Orientation Sectors

6.4.1.1 Oriented Gaussian and Linear Filters

The function g in each of the orientation sectors P_i has a different meaning, in the sense that the texture component v is derived differently from g in each of these regions. So special care has to be taken at the boundary between such regions. The g values from different sectors cannot be directly used in the same finite difference equation for the update of g at each iteration. Neumann conditions are not very easy to implement because of the irregular nature of the boundaries, and are not very accurate because of the fine detail involved with the oriented texture. Some type of inpainting [22], or extension of the isophotes across the boundary is necessary in order to solve the iterative Euler-Lagrange equations (Equations 6.2.10 and

6.2.11) with accuracy and stability.

Two different methods of extending the isophotes were attempted, one using oriented Gaussian filters (which will be seen to have the form given in Equation 6.4.1), and the other using linear angular filters (which will be seen to have the form of Equation 6.4.2). It was found that the oriented Gaussian filters were only modestly better than the linear angular filters, but much more inefficient. Therefore, the linear angular filters were used. A description of both types of filters now ensues.

For the oriented Gaussian filters, oriented diffusion of the g subcomponent is used, but adapted to be one-sided, so the only information used is from the same section as the pixel at which the solution is being calculated. According to [79, 86], such an oriented diffusion can be efficiently implemented by convolution with an appropriate oriented Gaussian filter.

Define the structure tensor $\mathbf{T} = \lambda_\eta \eta \eta^T + \lambda_\zeta \zeta \zeta^T$, where η is the gradient direction of f and ζ is the isophote direction. Then a corresponding oriented Gaussian can be defined as

$$G(\mathbf{x}, \mathbf{T}, \Delta t) = \frac{1}{4\pi\Delta t} \exp\left(-\frac{\mathbf{x}^T \mathbf{T}^{-1} \mathbf{x}}{4\Delta t}\right) \quad (6.4.1)$$

so that oriented diffusion after a time step Δt can be computed by

$$g^{t+\Delta t} = G(\mathbf{x}, \mathbf{T}, \Delta t) * g^t.$$

The amount of diffusion in each of the gradient and isophote directions can be controlled by the magnitude of the eigenvectors λ_η and λ_ζ of the structure tensor \mathbf{T} . Since diffusion is to be promoted mainly along the isophote direction, λ_η is

set to approximately 0 and λ_ζ to 1. Note that if λ_η is set exactly to zero, then the structure tensor is singular, and thus does not have a well-defined inverse. So λ_η is set equal to 0.01 in order to deal with this problem. An analytic expression for the inverse of the structure tensor was found using the symbolic computation package Maple [87], (not reproduced here), which was subsequently tested in the MATLAB implementation of OAD. This calculation of the tensor inverse was rather time consuming. Also, the oriented Gaussian in Equation 6.4.1 fades with distance from the filter centre, which is not desirable especially in early iterations, when pixels on the boundary have very low intensities in g . Therefore, linear angular filters are used for the convolution instead, these filters having the form

$$L(\mathbf{x}, \tau) = \delta_\epsilon(\sin(|\tau - \angle \mathbf{x}|)), \quad (6.4.2)$$

where τ is the estimated isophote angle at the pixel, $\angle \mathbf{x}$ is the angle that the spatial filter position \mathbf{x} makes with the filter centre, and δ_ϵ is the regularized Dirac delta-function given by the expression

$$\delta_\epsilon(s) = \frac{1}{\pi} \frac{\epsilon}{\epsilon^2 + s^2},$$

with ϵ a small positive constant (set to 0.1 for the experiments in this chapter). Note that the values of this filter only have angular dependencies and do not depend on distance from the filter centre.

6.4.1.2 Calculation of Isophote Angle of Linear Angular Filters

The isophote angle τ is estimated at each pixel which has been determined to have a phase discontinuity. This angle is computed by finding a weight for each neighbouring pixel in a window around the pixel with the phase discontinuity which satisfies a certain condition, which is now described. Call the pixel which has the phase discontinuity, p . Then weights are computed only for pixels in the 9×9 window around p which are from a different orientation sector or for which the gradient angle computed by the linear structure tensor as described earlier in Section 6.2.2 differs from that of p by at least 0.7π . For each such pixel q , the angle between q and p is found. Call this angle $\omega_1(p, q)$. The isophote angle of the window pixel is just $\frac{\pi}{2}$ plus the gradient angle, which was already computed by the linear structure tensor. Call this isophote angle $\omega_2(q)$.

Then the isophote angle τ of p is set to be equal to that of the isophote angle of the pixel in the window around p from the set of pixels that was considered having the maximal weight

$$w(q) = 2|\cos(\omega_1(p, q) - \omega_2(q))| + \frac{5}{2 + |f(p) - f(q)|}. \quad (6.4.3)$$

In the second term in Equation 6.4.3 above, $f(p)$ and $f(q)$ are the intensities of the original image f being denoised at pixels p and q respectively. This second term assures that pixels which are on the same isophote as that of p have higher weights. Note that this definition can be recursive in that isophote angles calculated for pixels with a phase discontinuity can be used to compute other isophote angles for pixels with phase discontinuities. The isophote angles τ are computed in row-

major order, and these are used as parameters to the linear angular filters of the form of Equation 6.4.2. These linear angular filters are then used to extend the isophotes of g , as will be described in Section 6.6.

6.5 Phase Unwrapping

The governing equation for θ , the angle component which is pre-computed in the OAD model of Section 6.2.1, is $\theta = \arctan\left(\frac{f_y}{f_x}\right)$. More simply, θ is the gradient direction of f . There is a distinction between direction and orientation. Two gradient vectors can point in opposite directions, but are said to have the same orientation, because if the vectors were to be drawn as arrows, but without their arrowheads, they would be exactly the same. In other words, two vectors have the same direction if they are the same modulo 2π , but they have the same orientation with a more relaxed condition, i.e. if they are the same modulo π .

Because $g_\theta = v$, it is desired to avoid π -jumps in θ , since this leads to instability in the result. To do this, phase unwrapping, which is used to reduce the number of or eliminate 2π -jumps, is performed. Phase unwrapping has applications to SAR interferometry, optics and magnetic resonance imaging. If Φ is an unwrapping function to unwrap 2π -jumps, then to eliminate spurious π -jumps, θ^{uwp} is set to $\theta^{uwp} = \frac{\Phi(2\theta)}{2}$. That is, the previous formula converts π -jumps to 2π -jumps, which are then partially eliminated and then division by 2 is performed since input angle was doubled before unwrapping. From now on, the function θ refers to θ^{uwp} , the orientation field with a reduced number of π -jumps.

To avoid unnecessary complication in code development, existing software was



Figure 6.4: fingerprint test image

used for the unwrapping procedure. There are two packages that were found on the World Wide Web to perform phase unwrapping. The first is Phase Vision [88], a commercial package with a limited-time evaluation download; the second is SNAPHU (Statistical-Cost, Network-Flow Algorithm for Phase Unwrapping) [89]. Although the unwrapped phase results obtained from Phase Vision were generally superior, the freely available SNAPHU also produced adequate results, and was used for the majority of the experiments. SNAPHU is based on a series of papers by Chen and Zebker, the first being [90], in which phase unwrapping is posed as an optimization problem solved using network flow techniques. As the full name suggests, SNAPHU relies on statistical models for phase unwrapping - in this case the statistical models are for SAR interferograms. Because the images tested here are more general, the statistical model is turned off by using a command-line option, and instead an L^1 -norm cost function is used.

For Phase Vision, a minimum-cost matching algorithm with branch cuts is used [91]; so like SNAPHU, a graph theoretical approach is taken.

Figure 6.5 shows the unwrapping process on a small section of the fingerprint



(a) Wrapped Angles for Section of fingerprint (b) Unwrapped Angles for Section of fingerprint

Figure 6.5: Example of Phase Unwrapping

image from Figure 6.4 - Figure 6.5(a) shows the wrapped angles while Figure 6.5(b) shows the result after unwrapping has been run using Phase Vision. Notice that in Figure 6.5(a), there are many neighboring arrows pointing in opposite directions. The unwrapped result in Figure 6.5(b) has all but eliminated these opposing arrows, but in general there may still be some π -jumps due to the intrinsic nature of the image. The phase unwrapping process was very fast with SNAPHU, usually complete in a fraction of one second.

6.6 Generation of Texture from g Subcomponent

As just stated, even after phase unwrapping, there may be some discontinuities in the form of π -jumps present in the phase. Also, there will be inherent phase discontinuities at the borders between the different orientation sectors. Thus these phase discontinuities have to be dealt with so that instability and incorrect solutions do not result. These phase discontinuities leads to incorrect reconstruction of

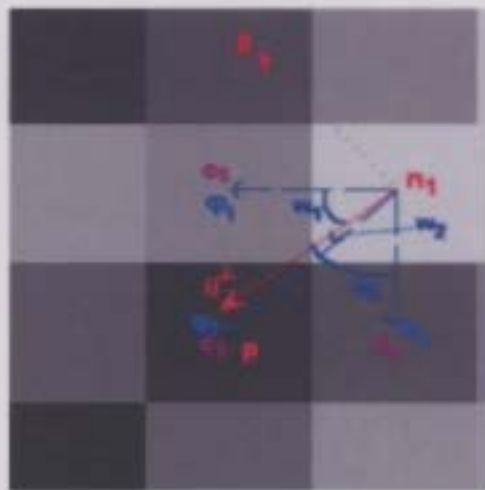


Figure 6.6: Generation of Texture Component from g at Phase Discontinuities

v from g in the form of very high positive and negative values at pixels where the phase discontinuities exist.

As will be seen in the denoising results section, Section 6.9.3.2, there are two categories of variations of the proposed OAD algorithm. One includes orientation sectors and rotation of derivative filters, while the other does not. First the generation of the texture component v from the g subcomponent for the simpler variation without orientation sectors is discussed, followed by an explanation of the generation of texture from g for the version with orientation sectors.

The first version of OAD published in [83] did not include derivative filter rotation or orientation sectors, and so included simpler adaptive adiabatic conditions using the information in g and θ . These boundary conditions were imposed by adaptively extending isophotes of g into the pixels with phase discontinuities. A description of how this was done now follows, and is shown diagrammatically in Figure 6.6.

There were no orientation sectors, and so a pixel was determined to have a phase

discontinuity only if one of its 4-connected neighbours differed in gradient angle by more than 0.7π . If a pixel did not have a phase discontinuity, the texture component v was calculated using the formula

$$v = g_\theta(i, j) = \cos(\theta(i, j))g_x(i, j) + \sin(\theta(i, j))g_y(i, j) = \cos(\theta(i, j))\frac{g(i+1, j) - g(i-1, j)}{2} + \sin(\theta(i, j))\frac{g(i, j+1) - g(i, j-1)}{2}.$$

At pixels with a phase discontinuity, all of the original g values in the 3×3 neighbourhood of the pixel should not be used in the finite-difference solution of the Euler-Lagrange equation for g . For example, if there is a π -jump at a pixel p , but the v component is smooth, then the g values around p will exhibit an abrupt sign change due to the fact that $g_\theta = v$ but there is a big jump in θ . For each of the 8 pixels $\{n_i\}_{i=1}^8$ in this neighbourhood, another 3×3 neighbourhood is formed, this time around n_i . The pixels in the neighbourhood around n_i are used to first determine the direction where the θ values are closer to the θ value for p . This direction will have angle approximately $\theta \pm \frac{\pi}{2}$. The three pixels c_1, c_2 and c_3 closest to forming an angle with p of $\theta - \frac{\pi}{2}$ in the 3×3 window around p , are considered and the sum $s_1 = \sum_{k=1}^3 \left(1 - \frac{|\theta(c_k) - \theta(n_i)|}{\pi}\right)$ taken. A similar sum is taken for those three pixels closest to forming angle with p of $\theta + \frac{\pi}{2}$. The maximum of the two sums determines which of $\theta \pm \frac{\pi}{2}$ is the isophote direction with more of the same θ values and thus the direction from which the g values with which to interpolate are taken. Call this isophote direction ψ_i . Each of the three closest pixels around n_i , say $\{c_k\}_{k=1}^3$, is at a certain corresponding angle $\{\phi_k\}_{k=1}^3$ to n_i . For example c_1 may be located one pixel to the right and one pixel to the left of n_i , in which case $\phi_1 = -\frac{\pi}{4}$. The

weights $\{w_k\}_{k=1}^3$ were formed by $w_k = (\phi_k - \psi_i)^2$. Then in the finite-difference calculation of $g(p)$, $g(n_i) = \sum_{k=1}^3 w_k g(c_k)$ is substituted for $g(n_i)$, which is in effect a weighted sum of g values in the isophote direction. Good results were obtained at a low cost using this approach, but recall that this method of extending isophotes is only valid for the version of OAD without orientation sectors and derivative filter rotation.

Now the generation of g for the more elaborate version of OAD, a slight modification of one already published [84], is presented, and includes the derivative filter rotation already described. For this method, the value of θ calculated from the linear structure tensor is examined to see in which of the orientation sectors, P_0, P_1, P_2 or P_3 it lies. If the angle θ is not in P_0 , then either $\pm \frac{\pi}{4}$ or $+\frac{\pi}{2}$ is added to θ to obtain a new angle θ_{adj} , which lies in P_0 . Then the rotated derivative filters, as found in Tables 6.1 and 6.2, along with θ_{adj} are used to compute the directional derivatives of u, f and g in Equations 6.2.10 and 6.2.11. So, for example,

$$g_\theta = \cos(\theta_{adj})(D_x * W_g) + \sin(\theta_{adj})(D_y * W_g).$$

W_g is a window of g values, of the same size as the filters D_x and D_y taken about the pixel r , for which the value g_θ is being calculated. The raw values of g are taken in W_g for those pixels which are in the same orientation sector as r . For those pixels in W_g that are not in the same orientation sector, or which differ in the calculated value of θ by at least 0.7π , an alternate computation of g is made.

For such pixels s , another window W'_g of g around the pixel s is taken. In W'_g , the g values of pixels which are in the same orientation sector as s are set to zero.

Then the value of g at s in the original window W_g is taken to be the convolution of the g values (some set to 0) in W'_g with the oriented angular filter of Equation 6.4.2 corresponding to the pixel s .

6.7 Non-Oriented Regions and How to Deal with Them

The proposed methodology works well when the underlying texture is oriented, but in other non-oriented regions tends to smear the noise, for which there is not a single dominant orientation. To avoid this, another decomposition model is needed to handle such regions. A region is defined to be non-oriented when its gradient direction coherence [55], as described in Section 2.7.2, is less than a pre-determined threshold. The energy functional that is used for such non-oriented regions is based on that of Bresson and Thiran [73], which was used in the context of applying decomposition to image segmentation. Their energy functional was

$$\mathbf{E}_{\text{BT}}(u, v) = \int_{\Omega} |\nabla u| dx dy + \mu \int_{\Omega} |v| dx dy + \lambda \int_{\Omega} (f - u - v)^2 dx dy \quad (6.7.1)$$

An extra term (with coefficient γ) is added to the energy of Bresson and Thiran, to obtain the energy shown below. The γ term has been added to promote noise being placed in the residual component r .

$$\begin{aligned} \mathbf{E}_{\text{NO}}(u, v) = & \int_{\Omega} |\nabla u| dx dy + \mu \int_{\Omega} |v| dx dy + \\ & \lambda \int_{\Omega} (f - u - v)^2 dx dy + \gamma \int_{\Omega} \phi(|\nabla v|) dx dy. \end{aligned} \quad (6.7.2)$$

Here $\phi(s)$ is a nonlinear diffusion potential, chosen to be $\phi(s) = 2\sqrt{1+s^2} - 2$, as in [92]. The Euler-Lagrange equations for this energy functional are then

$$\operatorname{div} \left(\frac{\nabla u}{|\nabla u|} \right) = 2\lambda(v + u - f)$$

$$\mu \operatorname{sign}(v) + 2\lambda(v + u - f) = \gamma \operatorname{div} \left(\frac{2}{\sqrt{1 + |\nabla v|^2}} \nabla v \right).$$

This particular potential $\phi(s)$ is chosen over others because it does not enhance strong edges, as Perona-Malik diffusion does for example [5]. Therefore it is better suited to the purposes of image decomposition, where the original image is the sum of cartoon, texture and residual components, and where such an enhancement in one component will necessarily appear as well in at least one other component. The important observation should also be made that now the texture component v is being directly dealt with, and not any subcomponents g .

Note that alternatively a subcomponent g could be used instead of v in non-oriented regions, with $g = v$, so that there would be one texture subcomponent throughout the image. In this case, the texture component v would be computed from the subcomponent g as

$$v = \begin{cases} \cos \theta g_x + \sin \theta g_y & \text{in oriented regions} \\ g & \text{in non-oriented regions} \end{cases} \quad (6.7.3)$$

6.8 Eikonal Orientation-Adaptive Decomposition

6.8.1 Derivation of EOAD model

In this section, the Eikonal Orientation-Adaptive Decomposition (EOAD) model is derived. The word eikonal is used in the name of this model because it has many similarities both in the definition of the texture component v in terms of the texture subcomponent g , and in the initialization of the texture subcomponent g , to the Eikonal partial differential equation $|\nabla F(x, y)| = P(x, y)$. The Eikonal equation was originally applied to a problem in the realm of image processing/-computer vision, namely shape-from-shading, in [93]. Using results from the OAD algorithm, it was found that in general $\theta \approx \arctan\left(\frac{f_y}{f_x}\right)$. It was found empirically from various runs of the OAD algorithm on oriented textured test images, the g -subcomponents were also oriented with the same gradient angles as f . Therefore, the relation $\theta \approx \arctan\left(\frac{g_y}{g_x}\right)$, holds as well. Thus, it follows that in most circumstances $g_\theta \approx \pm|\nabla g|$, since the gradient magnitude is equal to the absolute value of the maximal directional derivative of g . This relation can be exploited in the solution of the proposed model. However the ambiguity of the plus/minus sign is problematic. What makes this even more difficult to implement is that the relation $g_\theta = \pm|\nabla g|$ is only valid when there is exactly one strong orientation at a point, but is not valid when there is no oriented structure, or when there are two or more dominant orientations, e.g. at corners or triple-junctions. However, it is still possible to use the gradient magnitude based energy for strong-oriented regions, and another formula for less coherent regions. This would reduce the amount of computation necessary, because the angle related computations take up a substantial

part of the solution time of the OAD model.

As in Equation 6.2.9, for OAD the following relation was found to exist

$$f - u \approx \cos \theta g_x + \sin \theta g_y. \quad (6.8.1)$$

Since $\cos \theta g_x + \sin \theta g_y = \pm |\nabla g|$ (at least in oriented regions), there is the new constraint

$$f - u \approx \pm |\nabla g| \Rightarrow v = f - u \approx \text{sign}(f - u) |\nabla g|. \quad (6.8.2)$$

Converting this to a soft constraint, a new energy functional based on E_{OAD} in Equation 6.2.9 can be formed as

$$E_{\text{EOAD}}(u, g) = \int_{\Omega} |\nabla u| dx dy + \mu \int_{\Omega} |g| dx dy + \lambda \int_{\Omega} (|f - u| - |\nabla g|)^2 dx dy.$$

This energy functional is only valid in oriented regions where there is exactly one dominant orientation. The Euler-Lagrange equations from this energy functional, after some mathematical derivation and simplification, are

$$u - f + |\nabla g| \text{sign}(f - u) = \frac{1}{2\lambda} K(u) \quad (6.8.3)$$

$$\frac{\mu}{2\lambda} \text{sign}(g) = \nabla \cdot \left[\left(1 - \frac{|f - u|}{|\nabla g|} \right) \nabla g \right]. \quad (6.8.4)$$

6.8.2 Numerical Solution of Euler-Lagrange Equations and Implementation Details

The above Equations (6.8.3 and 6.8.4) are solved with a semi-implicit numerical scheme with a time-step $\Delta t = 0.01$. The initial conditions are a bit tricky to select, because if $g(t = 0) = g_0 = 0$, then it was found that g would stay close to 0 for all future times, as can be seen from Equation 6.8.4. For simplicity, and in conformance with all the previous decomposition models in this thesis, the initial condition $u(t = 0) = u_0 = f$ is used, with f the original image. For g , the Euler-Lagrange equation for u (Eqn. 6.8.3) is instead used to find initial conditions. Because $u_0 = f$, $|\nabla g| \text{sign}(f - u) = \frac{1}{2\lambda} K(u)$. Now, $f = u$, so in theory, the left-hand side should vanish, but if this is allowed to be perturbed slightly, then the Eikonal equation $|\nabla g| = |\frac{1}{2\lambda} K(u)|$ is obtained, since the gradient norm of g is always positive.

Here λ is chosen to be 0.05, and set to this value also for the first four iterations of decomposition. This Eikonal equation can be solved using the Fast Marching method of Sethian [94] in a matter of seconds. This was done in the tests here. The Fast Marching Toolbox for MATLAB [95], which includes very efficient MATLAB MEX code based on Sethian's Fast Marching method to solve this Eikonal equation, was used. In fact, it was found that solving the Eikonal equation with the right hand side $|\frac{1}{2\lambda} K(u_\sigma)|$, where u_σ is the cartoon component u blurred with a Gaussian filter of standard deviation σ (with σ small, say 1 or 2), gave superior results. It eliminated the need of detecting corners in the image and using a special PDE in a neighborhood around these corners. Corners are defined as being parts of the

image where there is more than one dominant gradient orientation. These can be found for example by the method in [96], which has the drawback of having a parameter κ which has to be determined empirically, or by detecting when both eigenvalues of the nonlinear structure tensor are distinct and large, which has the disadvantage of long running time. With the unblurred Eikonal equation, there was often instability at and in the vicinity of these corners. The Fast Marching algorithm propagates a level-set function from a number of starting points, where the solution is assumed to be 0, at a speed proportional to the right-hand side $P(x, y)$ of the Eikonal equation $|\nabla g| = P(x, y)$. The propagation is performed by a gradient descent of the distance function g , and finds an approximate geodesic (shortest path) through the image domain with pixel weights $1/P(x, y)$ using an algorithm similar to Dijkstra's algorithm for the shortest-path in a graph. The start points for the Eikonal equation solution were chosen to be all the points on the boundaries of coherent regions in the image.

Non-oriented regions are dealt with in the same way as for plain Orientation-Adaptive Decomposition, as described in Section 6.7.

6.8.3 Discussion of EOAD model

The EOAD model is a marked improvement over the regular OAD model because the calculation of the gradient angle of f has been removed completely. Only a very rough estimate of the angles from the noisy image f are used to determine which regions are coherent and which are not, but after this no angle information is utilized. Thus there is a significant time savings with the EOAD model, and

also somewhat more simplicity. No phase unwrapping or special calculation of the texture subcomponent g has to be made at pixels for which there is a discontinuity in gradient angle from those in neighboring pixels. The original divergence of two subcomponents g_1 and g_2 in the Vese-Osher decomposition model has been replaced by the magnitude of the gradient of one subcomponent g in oriented regions. More work needs to be done to relate this to function spaces, so that this work can be placed on a firm functional analytic footing.

It was found that the texture component v would become close to 0 as the decomposition progressed. Because theoretically the OAD and EOAD models should be equivalent, but this equivalence does not occur in practice, it is believed this is due to some intricacies in the implementation, e.g. the discretization of the Euler-Lagrange equations.

Better results were obtained by using the fact that θ is approximately the gradient angle of f , u and g in the Euler-Lagrange equations for OAD (Equations 6.2.10 and 6.2.11). However in that case the g -subcomponent obtained from solution of the model was very disordered and was not even oriented, although in tests conducted on the fingerprint image, the texture component v derived from g resembled that image quite closely. This is a matter which should be examined in future research.

6.9 Experimental Setup and Results

6.9.1 Test Images

The different versions of the proposed one-subcomponent OAD model are tested on six images: `fingerprint`, `marble`, `barbzoom`, `concentric`, `smallconcentric` and `eye`.

The test image `fingerprint` was chosen because it consists of oriented texture, along with a non-oriented background. The second test image `marble` was selected because it had more rapidly changing gradient angles, and had some very fine-scale oriented texture. The test image `barbzoom` was chosen since the background consists of some pixels where there is more than one dominant orientation. The two test images `concentric` and `smallconcentric` have oriented texture with all possible gradient angles, the former image with texture of larger scale than the latter. The final test image was `eye`, and was selected because it had a wide variety of both oriented and non-oriented texture.

The test image `fingerprint` is 160x192 pixels large, the test image `marble` has dimensions 210x287 pixels, `barbzoom` is 147x171 pixels in size, `concentric` is 200x200 pixels large, `smallconcentric` has dimensions 100x100 pixels, and `eye` is an angiogram of the eye of size 247x192 pixels. Both `concentric` and `smallconcentric` are synthetic images, generated using radial sinusoids of a fixed frequency emanating from the image centres.



(a) fingerprint



(b) marble



(c) barbzoo



(d) concentric



(e)
smallconcentric



(f) eye

Figure 6.7: Test Images used in Experiments

The equation for the image intensity function $f_1(x, y)$ of `smallconcentric` is

$$f_1(x, y) = \frac{1 + \sin(\sqrt{3\{(x - 50)^2 + (y - 50)^2\}})}{2}, \quad (6.9.1)$$

where x and y are integer pixel positions ranging from 1 to 100, except that $f_1(x, y)$ is set to zero within a distance of 7 pixels from the centre of the image.

Similarly, the equation for the image intensity function $f_2(x, y)$ of `concentric` is

$$f_2(x, y) = \frac{1 + \sin(\sqrt{0.5\{(x - 100)^2 + (y - 100)^2\}})}{2}, \quad (6.9.2)$$

with x and y integers ranging from 1 to 200, except that $f_2(x, y)$ is set to zero also within a distance of 7 pixels from the image centre. Both f_1 and f_2 are set to zero near their respective image centres to avoid the pathological case of having all orientations present at the image centres.

6.9.2 Experimental Setup

Different versions of the proposed OAD algorithm are tested for their ability to decompose and denoise the four test images. First, decomposition was run on the test images without added noise. Then in the second set of experiments for denoising, test images were generated by adding to each image white Gaussian noise of a rather high standard deviation $\sigma = 20$ to the original ‘noise-free’ images of Figure 6.7, and then decomposition was run with a dynamic value of λ based on the value of σ . In practice, if the standard deviation of the noise σ is not known, it can be estimated (see for example [97]). As stated earlier, most of the noise in the

OAD model will end up in the residual component r of the decomposition. The performance of OAD on the denoising of the `smallconcentric` test image, with three sets of filters was compared; the first set of filters was the more rotation-invariant derivative filters described in Section 6.3.1.1, the second set was the set of filters described in Section 6.3.1.2, first with filter rotation, and the third set was the set of filters in Section 6.3.1.2 without filter rotation.

Denoising based on Vese-Osher decomposition on the four test images was also run, and it was also seen there is more noise in the v component, and less in the residual. Vese-Osher decomposition is run in general for 10 iterations (less than that needed for decomposition, e.g. in Chapter 3, since we are now denoising), unless the SNR is still increasing at this point. In such a case, iterations are continued until the SNR (between the denoised and noise-free image) peak is achieved. A preliminary run of OAD is made, measuring the SNR at each iteration to find out at which point the peak in SNR between the uncorrupted image and the image denoised via decomposition is attained. Then each decomposition algorithm for the purposes of denoising is run up to the iteration with the SNR peak to obtain the timing results. For both algorithms and all images, μ is set to 0.05. The γ coefficient for non-oriented regions is set to a value of 0.2 for the denoising experiments, to ensure that noise is placed in the residual r . For the decomposition test runs, it is set to 0, since there is no added noise which has to be diffused. The fidelity parameter λ is chosen dynamically for both decomposition models, similar to the method in [38], since the noise variance is known. This is done in order to ensure unbiased comparison between the two decomposition algorithms. When the same parameters are chosen for both algorithms, the proposed algorithm performs rad-

ically better, but since the two models are different, these parameters should not generally be the same. As in [38], using the integral form of the known noise standard deviation σ and integration by parts on the Euler-Lagrange equations for u , an iteration-dependent choice of λ

$$\lambda = \frac{1}{2\sigma^2} \int_{\Omega} \frac{r_x u_x + r_y u_y}{\sqrt{u_x^2 + u_y^2}} dx dy,$$

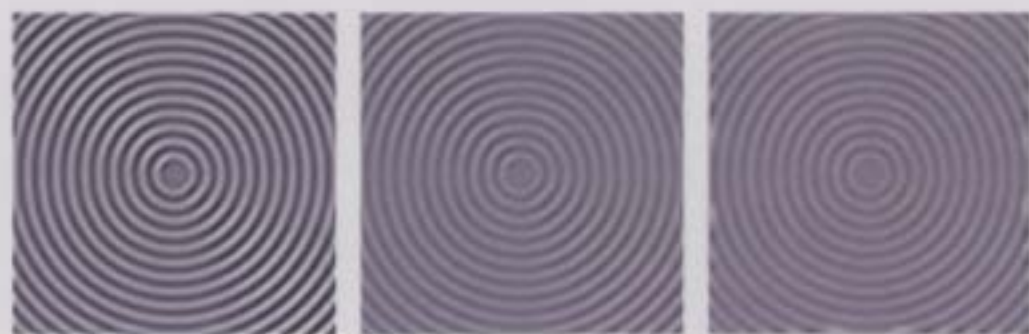
is arrived at for the OAD algorithm. Here r is the residual as earlier.

Finally, OAD requires an image-dependent orientation coherence threshold to determine which regions of the image are oriented. This threshold was set to the following values for the test images in this chapter: 30 for fingerprint, 15 for barbzoo, 13 for marble, 20 for eye and 10 for smallconcentric. The overall OAD method was not very sensitive to these values, which were chosen to roughly correspond to what an observer would consider to be the oriented and non-oriented regions of the image being decomposed or denoised. As already stated in Section 6.2.2, the linear structure tensor is used for orientation estimation.

6.9.3 Results

6.9.3.1 OAD Results for Decomposition

In Figure 6.8, the effect of derivative filter rotation on decomposition (not denoising) results for the test image concentric is examined, as well as a comparison to Vese-Osher decomposition, on which OAD is based, made. Observe that for some angles in the texture component of OAD decomposition without derivative



(a) Texture Component v of V-O Decomposition (b) Texture Component v of OAD Decomposition without Derivative Filter Rotation (c) Texture component v of OAD Decomposition with Derivative Filter Rotation

Figure 6.8: Comparison of Texture Components of Decomposition from V-O Algorithm and OAD *with* and *without* Derivative Filter Rotation

filter rotation in Figure 6.8(b) are more prominent than others. This orientation-dependent effect is also seen for the texture component from the V-O decomposition of *concentric* in Figure 6.8(a). However, in Figure 6.8(c), where the texture component v of OAD with derivative filter rotation is shown, this effect is much less pronounced. This illustrates one advantage of the proposed OAD scheme, in addition to the simplicity of having one texture subcomponent instead of the usual two.

In Figure 6.9, the result of running OAD with filter rotation and with the optimized derivative filters from Section 6.3.1.1 against the V-O decomposition result are compared on a real-world image, *fingerprint*. OAD is run for 10 iterations while V-O decomposition is run for 9 iterations (this is the first iteration where the total variation of the cartoon component u is less than that from OAD after 10 iterations) to see how the components appear for both models. For both algorithms, the fidelity parameter λ was set to 0.02 and the parameter μ was set to 0.05. Usually the two subcomponents for Vese-Osher decomposition have initial conditions

$g_1 = -\frac{1}{2\lambda} \frac{f_x}{|\nabla f|}$ and $g_2 = -\frac{1}{2\lambda} \frac{f_y}{|\nabla f|}$, but to allow fair comparison with OAD, the initial conditions $g_1 = g_2 = 0$ are used, since for OAD, the initial condition for the texture subcomponent g is $g = 0$. Note that the usual initial conditions for V-O decomposition were used in the previous V-O decomposition example in Figure 6.8(a).

Vese-Osher decomposition did a slightly better job of extracting the fingerprint texture from the original fingerprint image (the texture component in Figure 6.9(d) is more pronounced than that in Figure 6.9(c)), but OAD still performed well. There is more orientation selectivity with the Vese-Osher decomposition result than the OAD result, as some parts of the fingerprint in the texture component of the V-O decomposition are brighter than others, though there is not a substantial difference in their scale or in their intensity in the original fingerprint image. The L^2 -norms for the non-cartoon components of the two decompositions were: 4.34×10^3 for the v component from V-O decomposition vs. 3.37×10^3 for the v component from OAD, and 1.53×10^3 for the r component from V-O decomposition vs. 2.11×10^3 for the r component from OAD. The decrease in texture extraction capability is thought to be due to small inaccuracies in gradient angle estimation in OAD, as well as discretization error.

A similar experiment was performed for the `smallconcentric` image for which the difference between the texture extraction performance of V-O decomposition and OAD was not as pronounced. For that image, the L^2 -norm of the v component was 5.88×10^3 for V-O decomposition vs. 4.85×10^3 for OAD, and the L^2 -norm of the residual component r was 1.97×10^3 for V-O decomposition vs. 2.36×10^3 for OAD, after 10 iterations of each algorithm. This improved texture extraction is thought



(a) Cartoon component u of OAD3 Decomposition (b) Cartoon component u of V-O Decomposition



(c) Texture component v of OAD3 Decomposition (d) Texture component v of V-O Decomposition



(e) Residual component r of OAD3 Decomposition (f) Residual component r of V-O Decomposition

Figure 6.9: Comparison of Decomposition Results (without added input noise) from V-O and OAD on fingerprint with Zero Initial Condition for Texture Components

to be due to the smaller scale of texture present in `smallconcentric` as opposed to `fingerprint`.

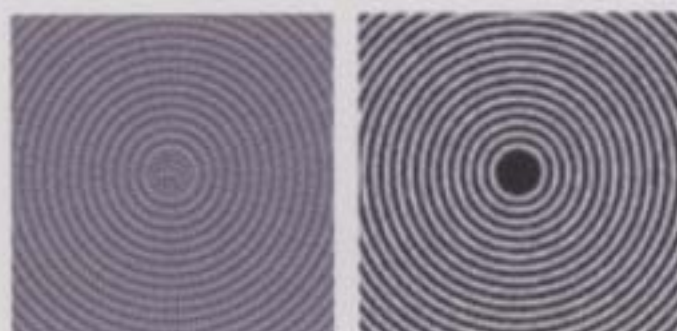
6.9.3.2 OAD Denoising Results

Next the denoising performances of OAD and V-O denoising are compared, where it is seen that OAD does a better job at denoising than V-O decomposition. First of all, it is seen once again as talked about in Section 6.3.1.2, that filter rotation is necessary in the OAD algorithm for the case of denoising. It was found that OAD denoising was most effective for small scale texture, and for large-scale texture there was often noise left in the texture component v . This is, however, precisely the kind of texture as seen in Chapter 5, that OSV denoising performs poorly on, e.g. the `barbara` test image from that chapter. Results are shown here for three sets of runs on the `smallconcentric` image - first for denoising with the same rotation-invariant derivative filters as for decomposition (which is called OAD1), second with the specialized filters based on central differences (non rotation-invariant) from Section 6.3.1.2, and without filter rotation (which is called OAD2), and then third with the same specialized filters with filter rotation (called OAD3). If only OAD is used as an abbreviation, then this refers to OAD3, since this generally gave the best denoising results out of the three versions of Orientation-Adaptive Decomposition.

For OAD1, where the same rotation-invariant filters as those based on the ones found in Equations 6.3.2 and 6.3.3 are used, results for the `smallconcentric` image are presented in Figure 6.10. As can clearly be seen, the OAD model with these filters is not suitable for denoising. For instance, the residual component r in Figure



(a) Original (b) Cartoon Component (c) Texture Component v
 smallconcentric im- u of OAD decomposition of OAD Decomposition
 age with Added Noise with Derivative Filter with Derivative Filter
 ($\sigma = 20$) Rotation and Rotation- Rotation and Rotation-
 Invariant Filters Invariant Filters



(d) Residual Component (e) Denoised Result $u + v$
 r of OAD Decomposition of OAD Decomposition
 with Derivative Filter with Derivative Filter
 Rotation and Rotation- Rotation and Rotation-
 Invariant Filters Invariant Filters

Figure 6.10: The Components of the OAD Decomposition of smallconcentric and Final Denoised Result *with Filter Rotation* and *with Rotation-Invariant Derivative Filters* (10 iterations, OAD1)

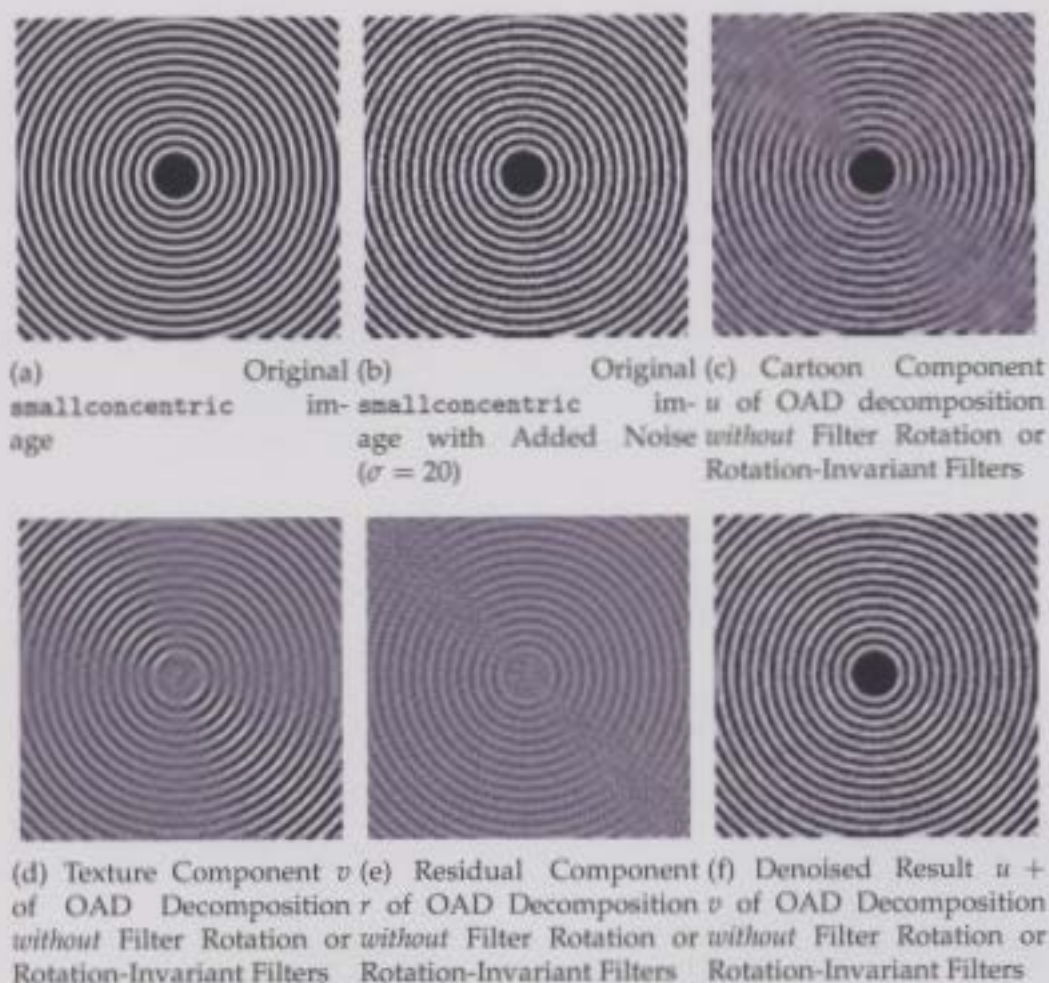


Figure 6.11: The Components of the OAD Decomposition of *smallconcentric* and Final Denoised Result *without Filter Rotation* and *without Rotation-Invariant Derivative Filters* (10 iterations, OAD2)

6.10(d) contains a very large amount of the circle texture, much more than the noise in this component.

In Figure 6.11 is seen the 3 components u , v and r of the OAD2 decomposition result on the test image `smallconcentric`. This time, the filters have been changed, but filter rotation is not performed. As with the decomposition, there is more strength in the texture component v for angles θ with $\tan(\theta)$ close to -1. For such angles, the denoising performance of OAD2 is also very good, since at these angles, the residual r consists almost exclusively of noise. However for other angles, there was a large amount of texture transferred to the residual component r , although this component should be reserved for noise.

The denoising results of Figure 6.12 correspond to OAD3, and are the best of the OAD results on the `smallconcentric` image. In the limited range of angles with tangent close to -1, there is a diminution of denoising quality as compared to OAD2, but this effect is very slight, and for other angles, the OAD3 denoising results are much better than the OAD2 ones.

Overall, for `smallconcentric`, the SNR values of the denoised result ($u + v$) for the OAD implementations after 10 iterations of OAD1 and OAD2, and 12 iterations of OAD3, were: 11.88 for OAD1, 12.02 for OAD2 and 15.19 for OAD3. The SNRs of OAD1 and OAD2 were very close (within 0.2) to each other, and OAD3 was the clear winner of the group of OAD implementations, at least for this test image. Since this test image contains all orientations, OAD3 is chosen as the standard OAD implementation for texture denoising.

Denoising results based on OSV decomposition and V-O decomposition are also

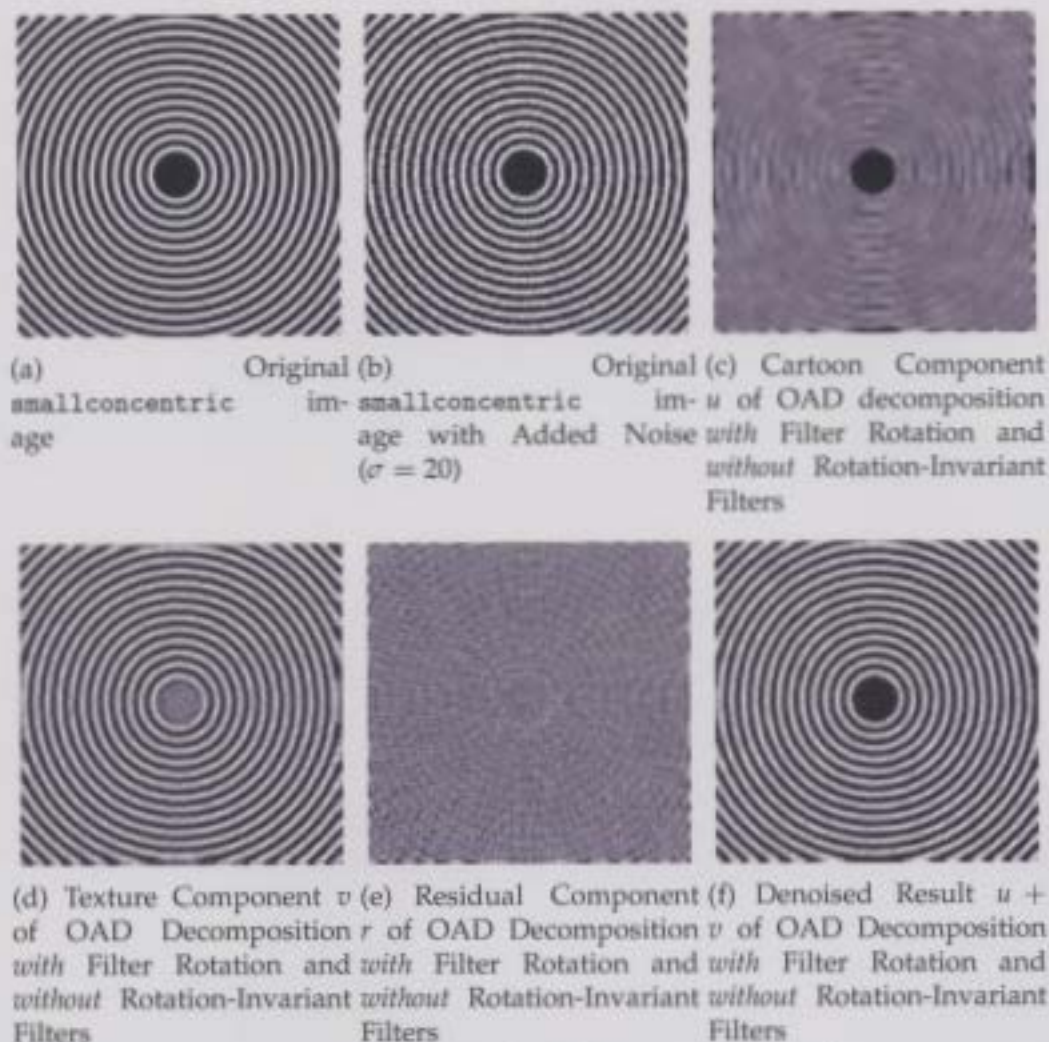
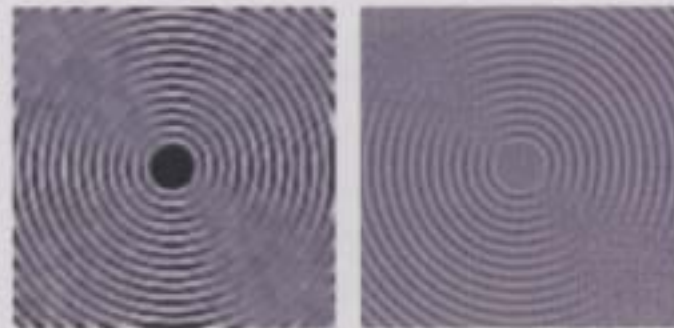


Figure 6.12: The Components of the OAD Decomposition of *smallconcentric* and Final Denoised Result *with* Filter Rotation and *without* Rotation-Invariant Filters (12 Iterations, OAD3)



(a) Cartoon Component u of OSV decomposition (b) Noise Component v of OSV decomposition



(c) Cartoon Component u of V-O decomposition (d) Noise Component r of V-O decomposition

Figure 6.13: Denoising Results on *smallconcentric* based on OSV and V-O decomposition models

compared with OAD results. In Figures 6.13(a) and 6.13(b), it is seen that very little noise gets transferred to the v component of the OSV decomposition before the SNR of the denoised result goes down. The final SNR for OSV denoising on `smallconcentric` is 13.25, which is less than that for OAD, and this OSV result is achieved in only 25 iterations. In the first several iterations, the SNR of the denoised result from V-O denoising is quite high, but steadily decreases. In these early iterations, there are results similar to that from OSV denoising, where there is very little energy in the noise component. After 22 iterations, another SNR peak is reached, this time with an SNR of 11.40, and the results after 22 iterations are shown in Figure 6.13(c) and 6.13(d). The noise component r is comparable to that of OAD in that the best denoising performance is obtained for a narrow range of angles θ , with $\tan(\theta)$ close to -1. However the quality of denoising from OAD is better than that from V-O, as can be seen from visual comparison of Figures 6.13(d) and 6.11(e), and will soon be seen in numerical comparisons in Table 6.3.

OAD was generally the best denoising algorithm on the test images considered in this chapter, which is remarkable since it is based on the V-O model, and still outperforms the OSV model, which is designed for denoising while the V-O model is not. This performance improvement can be seen in Table 6.3 for the five test images.

This suggests that a model similar to OAD but based on OSV decomposition instead of Vese-Osher decomposition would give even better results than OSV decomposition when used for denoising. This modification however would be different than the models in Chapter 5. There was only a deterioration in denoising performance for OAD as compared to OSV denoising for the `barbzoom` image. This

Test Image	Signal-to-Noise Ratio of Denoised Images		
	V-O denoising	OAD denoising	OSV denoising
fingerprint	9.99	12.41	11.57
barbzoom	6.46	9.82	9.93
marble	5.32	7.71	7.64
eye	12.99	15.32	14.98
smallconcentric	11.51	15.19	13.25

Table 6.3: Signal-to-Noise Ratio (SNR) of OAD denoising vs. V-O and OSV denoising results

is to be expected since most of the background consists of texture where there are two dominant orientations, and thus the estimation of gradient orientation with the structure tensor is not accurate. This problem could perhaps be circumvented by selecting a larger window in the computation of gradient direction coherence (Section 2.7.2) to compute which regions are oriented, since with the smaller 7 by 7 pixel window used throughout this thesis, and in the experiments in this chapter, in particular, some pixels with more than one dominant gradient direction were still considered to be in the oriented part of the image. However, the difference in SNR was so insignificant in this image (9.82 for OAD vs. 9.93 for OSV), that this is not a major issue.

Figure 6.14 shows Vese-Osher (V-O) image denoising results on fingerprint. Observe that there is still quite a bit of noise in the texture component v and that there is a lot of texture in the residual r . In Figure 6.14(c), a zoom of the sum of cartoon and texture components is shown, corresponding to $f - r$. In this context, this corresponds to the denoised image. In Figure 6.15, it is seen that the decomposition results for the OAD2 version of the Orientation-Adaptive model are also good. There is less noise in the texture component v and the residual r consists of more noise than for V-O decomposition. Quantitatively, the SNR of the denoised result

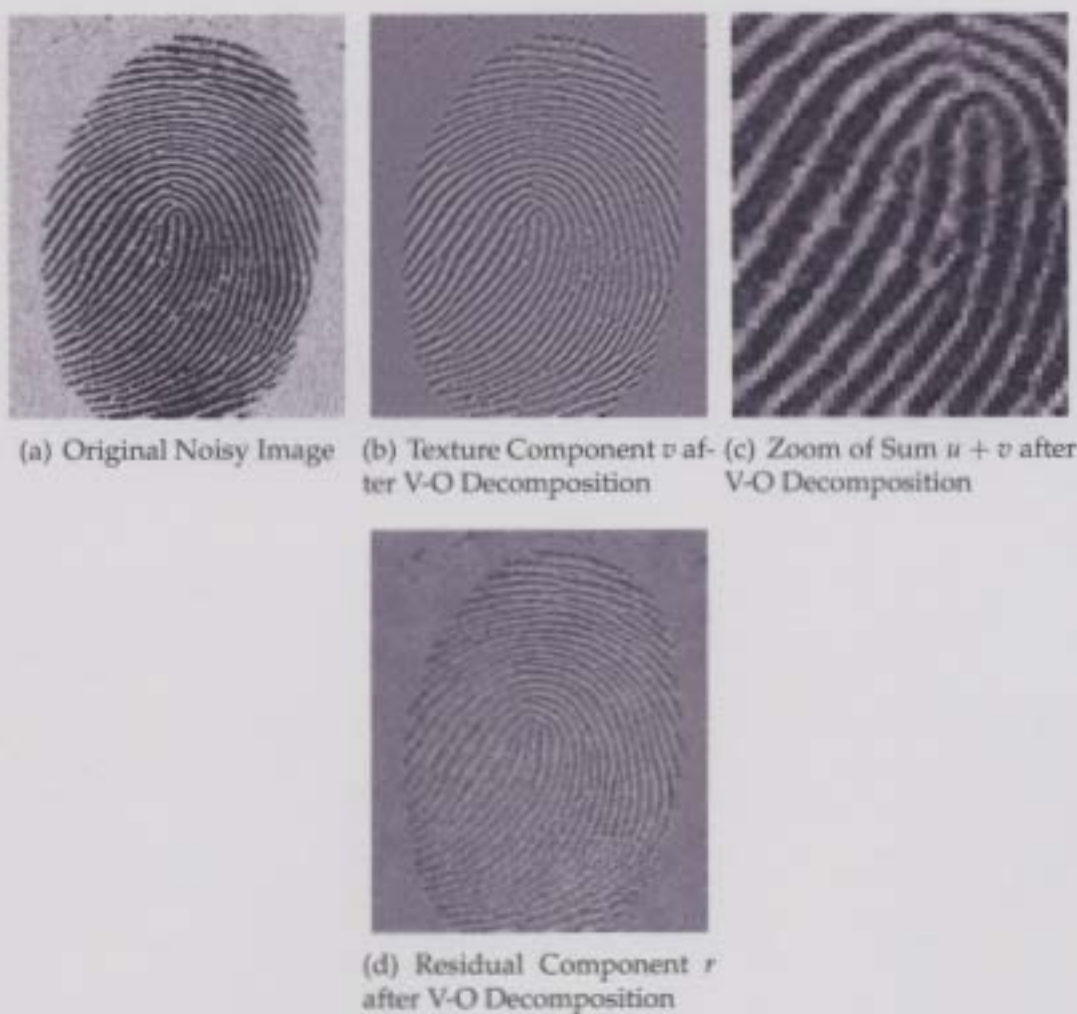


Figure 6.14: Various components from V-O decomposition for denoising of noisy fingerprint

from the proposed algorithm without filter rotation and with rotation-invariant derivative filters (OAD2) is 11.07, while for V-O decomposition, this SNR is 9.99. Once again, in Figure 6.15(b), a zoom of the sum $u + v$, a roughly denoised image, is shown.

The OAD3 algorithm was also run on fingerprint, in which the central difference-based filters of Section 6.3.1.2 were used, with filter rotation. After 9 iterations, this gave an SNR of 12.41. V-O denoising gave a much lower SNR of 9.99. As seen in Figure 6.16(c), the residual contains less texture than the residual for OAD2 and V-O denoising. As well, the amount of noise is more uniform across all gradient angles, as it can be seen that there is approximately an equal amount of texture in the v component in Figure 6.16(a) across the entire fingerprint, which is not what is observed for the OAD2 result in Figure 6.15(a).

In Figure 6.17, the noisy image eye, is also denoised with the proposed OAD3 algorithm (by taking $u + v$). Though this is strictly not an oriented texture, the use of directional diffusion in the proposed model allows this image also to be denoised.

There is less success on the barbzoo image because at many of the pixels in the image there is more than one dominant orientation. Thus instability occurs, and a relatively bad denoising result ensues, as seen in Figure 6.18. There is quite a bit of texture in the noise component r (not shown), and the beginning of the aforementioned instability was visible in this component. Despite this, OAD3 still outperforms Vese-Osher denoising.

The timing results of the three denoising algorithms as well as the number of

iterations each algorithm is run are shown in Table 6.4. OAD denoising was run until the SNR peak was achieved for all test images being denoised. V-O denoising was also run until an SNR peak occurred between the denoised result and the image uncorrupted by noise, or after 10 iterations, whichever came later. This latter condition was included because often multiple SNR peaks would occur, some before the absolute SNR peak was achieved. Finally, OSV denoising was run until the SNR peak was attained.

The timings for OAD were one or two orders of magnitude greater than those for the other algorithms. The code in its present form is not entirely optimized for speed; with additional work, the running times should be made substantially lower. Of the two benchmark decomposition algorithms, Vese-Osher and Osher-Solé-Vese decomposition, which are applied to denoising and which are compared to Orientation-Adaptive Decomposition, only Vese-Osher decomposition splits the original image into the sum of cartoon, texture and noise components. OSV decomposition on the other hand, although being the only decomposition algorithm competing with OAD yielding denoising results of comparable quality, only gives two components, the denoised component u and the noise component v , when applied to the problem of denoising. Therefore, the additional time complexity of OAD can be reconciled with the fact that it is doing more, by both decomposing and denoising the original image. However the additional time required by OAD could become an issue for real-time applications, which are beyond the scope of the research in this thesis.

Test Image	Time (sec) for Denoising			No. of iterations		
	V-O	OAD	OSV	V-O	OAD	OSV
fingerprint	13.97	1114.08	32.80	25	9	466
barbzoom	4.41	572.67	47.41	10	5	694
marble	25.20	1535.25	187.30	22	6	1112
eye	3.31	1295.31	114.73	4	6	894
smallconcentric	1.75	721.01	0.44	12	10	25

Table 6.4: Execution Time and Number of Iterations Required for OAD Denoising vs. V-O and OSV Denoising



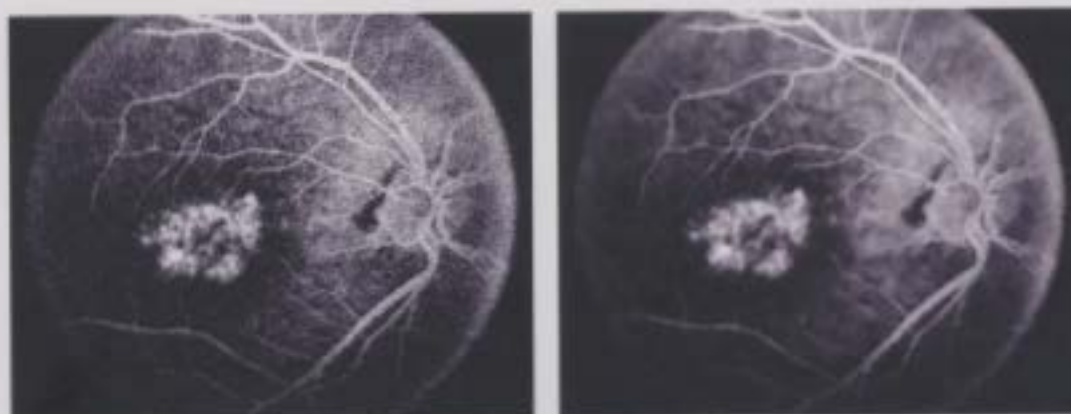
(a) Texture Component v (b) Zoom of Sum $u + v$ after (c) Residual Component r
after Proposed Decomposition Proposed Decomposition after Proposed Decomposition

Figure 6.15: Various components from proposed Orientation-Adaptive Decomposition (OAD2) of noisy fingerprint



(a) Texture Component v after Proposed Decomposition (b) Zoom of Sum $u + v$ after Proposed Decomposition (c) Residual Component r after Proposed Decomposition

Figure 6.16: Various components from proposed Orientation-Adaptive Decomposition (OAD3, 9 iterations) of noisy fingerprint



(a) Original noisy eye image

(b) Denoised eye image with proposed OAD3 algorithm

Figure 6.17: Denoising results with Orientation-Adaptive Decomposition (OAD3, 6 iterations) on noisy eye



(a) Original noisy barbzoom image (b) Denoised barbzoom image with proposed OAD3 algorithm (c) Denoised barbzoom image with V-O decomposition

Figure 6.18: Denoising results with Orientation-Adaptive Decomposition (OAD3, 5 iterations) and Vese-Osher decomposition on noisy barbzoom

6.10 Conclusions

In this chapter, a one-subcomponent textured image decomposition model, called Orientation-Adaptive Decomposition, was proposed. This model was applied to the decomposition and denoising of oriented texture. Although there are other methods for the denoising of oriented texture, the proposed method stands out because it is based on a decomposition model, and thus the cartoon and texture components of the obtained decomposition can be used for prioritized image transmission or image inpainting, if some parts of the region are damaged. The OAD model was also extended to a simpler and more elegant model not requiring as many computations called Eikonal Orientation-Adaptive Decomposition (EOAD). However, more work is needed in the implementation to make EOAD a viable alternative to ordinary OAD.

OAD, like the OLOSV model of the previous chapter is an effective one for the

denoising of oriented texture. The main difference is that it is also very effective for the problem of normal image decomposition without noise, and is a novel model for this problem, also computing cartoon and texture components for the denoising case.

Conclusions

7.1 Thesis Summary

In this thesis, the problem of image decomposition has been studied from a variational point of view, with existing decomposition models extended and new models created. The models described and developed in this thesis improve the quality of decomposition and in many cases the efficiency as well.

In general, existing decomposition models were modified either by adding extra terms to their defining energy functionals under the variational framework, or by modifying existing terms, e.g. by introducing local image-dependent rotations. The addition of extra terms provided some regularization of the solutions of the PDEs, and also imposed some constraints between the various functions in the defining energy functional of the various models. With both the existing and novel decomposition models, care had to be taken with their solution, which generally involved the numerical solution of a system of partial differential equations

derived from the model's energy. Solutions were implemented with either explicit or semi-implicit methods, the latter usually being more stable. Additive Operator Splitting [36] was seen to be a particularly fast implementation of nonlinear diffusion flows.

Chapter 1 consisted of an introduction to image processing and its current open problems. Then, the motivation for the use of image decomposition in the solution of these problems and an overview of existing variational image decomposition models were given in Chapter 2.

Improved Edge Segregation (IES), a new model created based on the existing decomposition model by Vese and Osher was introduced in Chapter 3, with results on both serial and parallel computers given. The decomposition results from IES were found to be better than those from the Vese-Osher model, as measured by the lack of cartoon edges in the obtained texture component. Additionally, the decompositions were found in less time than previous models, due in large part to the fact that IES was found empirically to be implementable with Jacobi iterations as opposed to Gauss-Seidel iterations. This difference was found to be even greater on a multi-computer cluster because of the increased capacity for parallelization of the solution. As well, the output from IES was shown to give superior discrimination results when fed into an Active Contours without Edges scheme as compared to the output from Vese and Osher's algorithm.

In Chapter 4, another new variational image processing model, Simultaneous Decomposition/Discrimination (SDD), was developed. As the name suggests, it is used for the simultaneous discrimination and decomposition of textured images, combining the two stages in the discrimination tests in Chapter 3 into one. This

model used the natural coupling of the two problems where information from one could be used for the solution of the other, leading to more effective solutions for both.

In Chapter 5, two extensions were presented to the decomposition model of Osher, Solé and Vese (the OSV model), one which decorrelated the cartoon and texture components of an image, and the other which combined two frameworks, 1) decomposition and 2) nonlinear diffusion. The first extension, the DOSV model, was shown like IES to more accurately separate cartoon and texture information into their respective components. For the second extension to OSV, the new models, incorporating the Perona-Malik and Oriented-Laplacian nonlinear diffusion schemes, respectively, combined both decomposition and nonlinear diffusion and were used for the purpose of textured image denoising. The latter nonlinear diffusion model was especially effective in preserving many textured regions in the cartoon component u and not in what is now defined to be the noise component v . Denoising results were better than those obtained by the plain OSV model.

A new model for the decomposition of oriented texture, called Orientation-Adaptive Decomposition (OAD), was brought forward in Chapter 6 and this model was applied to the denoising of such textures. This model reduced the number of texture subcomponents from two to one, and relied on the pre-calculation of orientation in the original image. A variation of OAD called Eikonal Orientation-Adaptive Decomposition was also presented. The advantage of this scheme was that it did not require the precomputation of orientations as does OAD, leading to a time savings.

7.2 Future Work

In the introductory Chapter 1, it was stated that the aim of this thesis was to modify existing and create new variational decomposition models in order to represent digital images as a sum of components, in order to facilitate the solution of other image processing and analysis problems. With the work presented in this thesis, the quality and efficiency of image decomposition schemes have been improved so that they could possibly be applied to high-quality real-time and hardware implementations of decomposition. Though real-time and hardware implementations were not done for this thesis, it is a natural and logical step to be explored further in the future. Especially for larger sized images, the efficiency of decomposition becomes an issue, and it is also important if image decomposition is to be applied to image sequences, e.g. for the automatic registration of such sequences [23]. A comparison of IES with V-O decomposition in this regard would be especially interesting to see. The statement about the importance of efficiency is also valid for colour/multispectral images, because they have more than the usual one colour intensity plane present in grayscale images, and so require the calculation of more than one value per component per pixel.

In Chapter 3, the IES model was implemented on a Beowulf cluster with only a handful of nodes, and a substantial speed-up was witnessed. Implementation on a much larger parallel computer to see how fast it could be made is a logical next step. Based on the results obtained on the small cluster in this thesis, such decomposition could be performed in a fraction of a second, given enough processing power. Additionally, the other models in this thesis should be implemented in a

parallel fashion using MPI or another high level parallel language to see how much they could also be sped up. As well from Chapter 3, more work should be done on the Geometric Constraint model, either by combining it with the IES model or developing it further on its own, so that convergence is improved.

For the SDD model of Chapter 4, a more systematic justification and setting of parameters should be sought, along with different numerical methods for the solution of the model which are more easily proven to converge to a global minimum of its defining energy functional. The SDD model is based on Vese-Osher decomposition, and also compared to sequential decomposition/discrimination with Vese-Osher decomposition as the first step. It would be useful to instead use the IES decomposition model, also proposed in this thesis, as the basis of and comparison against, the SDD model. Better results would be expected, since in Chapter 3 better discrimination results were obtained by IES over V-O when applied sequentially as the first step, followed by Active Contours Without Edges discrimination.

The decomposition/discrimination algorithms should also be tested on noisy images to see how robust they are to corruption by noise. It is suspected that the model will be able to withstand small amounts of noise, especially for the discrimination phase, because of the Gaussian blurring of the $|g_i|$ subcomponent. With larger amounts of noise, much of the $|g_i|$ subcomponent will consist of the noise itself, and there will be less separation between the textures in the channel consisting of that subcomponent for the purposes of discrimination.

Though decomposition was applied to several problems, such as texture discrimination and denoising, the new and existing decomposition models have not been applied to the problem of image inpainting. It would be interesting to see if similar

ideas to those used for the simultaneous decomposition/discrimination scheme in Chapter 4 could be used for the simultaneous solution of decomposition and inpainting, or even the simultaneous solution of all three problems - decomposition, discrimination and inpainting.

The OLOSV model of Chapter 5 could be extended to images with multiple dominant orientations [98]. Additional work is also needed to convert it to a model for pure decomposition, rather than primarily for denoising, as it is now.

There are many further modifications that can be made to the OAD and EOAD models of Chapter 6. For future work, more care can be taken with the approximations, both in the derivations of the OAD and EOAD models, as well as with the implementation, perhaps improving the obtained results. An alternative model could be formed by using closed-form expressions in terms of the partial derivatives of the various components/subcomponents of the directional derivatives of these components/subcomponents in the Euler-Lagrange equations from the OAD model, and not in the energy. This was not done here, but gave promising results when attempted, and is left to be completed for future work.

It is desired to make the OAD denoising algorithm blind by roughly estimating noise variance [97] so that the standard deviation does not have to be known beforehand. If this were to be done, then it would have to be ensured that OAD is robust to errors in the noise variation estimation. More direct comparisons should be made between OAD and OLOSV denoising in the future to see which of the two proposed denoising algorithms is better for which images. Automatic parameter selection for the energy coefficients of the OAD model is desirable, and this work should be extended to colour images, as with other methods in this thesis.

Additionally, a better choice of derivative filters and general numerical implementation may improve the practicality of using EOAD for denoising, and perhaps the restriction of the necessity of having an image with some oriented texture as input could be relaxed.

In fact, of the proposed decomposition models in this thesis, OLOSV, OAD and EOAD all require that there at least be some strongly and singly-oriented regions in the image that the model is being applied to. Otherwise the proposed model would degenerate to another one (OSV for OLOSV, and Bresson and Thieran's model [73] for OAD and EOAD). Occasionally, as with the *barbzoom* test image in Table 6.3, where OSV gives a slightly better denoising Signal-to-Noise Ratio, the proposed decomposition models may not provide optimal performance. In this case, it could be beneficial to use an image classification scheme to determine which decomposition model should be applied to any given image to give the best decomposition/denoising quality. Such a classification scheme, which is left to future work, could be partially based on what fraction of an image consists of oriented structure, among possibly other image features.

As well, more effort could be made to combine the new models in this thesis. If this were to be done, it is expected that results would be even better, except that perhaps efficiency would be sacrificed. For example, the IES model of Chapter 3 could be incorporated in the OAD model of Chapter 6 to remove cartoon edges from the texture component, and a version of IES for a two component model could be used in DOSV from Chapter 5 to further improve performance over existing models. OLOSV from Chapter 5 and OAD from Chapter 6 could also be combined to provide better denoising performance for oriented texture images.

Finally, the decomposition models in this thesis were not applied to colour images. This is a definite future step that should be taken, and could be implemented for all the models in this thesis. This could be done using the colour Total Variation model of Chan and Blomgren [99]. Previous instances of colour decomposition models in the literature are [100] and [101]. Similar ideas to those used in those papers could be carried over to the decomposition models in this thesis.

7.3 Final Reflections

Image decomposition has become important in the image processing research community only in the last several years. However, it is a fundamental problem in the field as it reaches to the deepest image processing questions, such as "What is Texture?" and "What is Noise?" for which there are no universally accepted precise answers. Though no attempt to answer these types of questions were made in this thesis, ideas related to these questions often played a role in the development of the algorithms herein.

It is believed that the representation of images by components from variational decomposition models will be a central part of many image processing applications in the coming months and years, for example for PDE-based compression [102]. With efficient algorithms to perform decompositions of high quality, it is more likely that researchers and other practitioners will use decomposition frequently when developing their own image processing solutions, for example in the way that the Fourier Transform or wavelets are used today. It is hoped that this thesis will help these researchers in achieving their goals.

Analytical Solution of OSV Decomposition Model

A.1 Introducing Subcomponents into the OSV Decomposition Model

In the Vese-Osher model, there are two subcomponents g_1 and g_2 , the absolute value of one of which can be used for the purposes of discrimination. However, in the Osher-Solé-Vese model, the subcomponents are eliminated in the Hodge Decomposition used for its solution, so that discrimination is no longer possible directly with these subcomponents. Instead there is just the cartoon component u , and the texture is whatever is left over, $f - u$. However, it is possible to preserve the presence of these subcomponents if hard constraints are used to enforce that the texture $v = \operatorname{div}(g_1, g_2)$. In the OSV model, the square of the Sobolev H^{-1} -norm of v is included in the energy functional. It is now shown that the same

evolution equations for u can be derived using a much simpler approach, that maintains the subcomponents g_1 and g_2 which can in turn be used for the purposes of discrimination.

Consider the modified Vese-Osher functional, where instead of using the L^∞ -norm of $\sqrt{g_1^2 + g_2^2}$, the square of the L^2 -norm of $\sqrt{g_1^2 + g_2^2}$ is used. This is done to simplify the subsequent derivation. The following functional is hence obtained

$$\begin{aligned} E_{\text{MVO}}(u, g_1, g_2) = & \int_{\Omega} |\nabla u| dx dy + \lambda \int_{\Omega} (f - u - g_{1,x} - g_{2,y})^2 dx dy \\ & + \mu \int_{\Omega} (g_1^2 + g_2^2) dx dy. \end{aligned} \quad (\text{A.1.1})$$

Instead of including the middle term with coefficient λ , which is a soft constraint ensuring that $f \approx u + v \approx u + g_{1,x} + g_{2,y}$, a hard constraint is included, so that a new energy functional to be minimized is obtained

$$E_{\text{MVO}^{\text{con}}}(u, g_1, g_2) = \int_{\Omega} |\nabla u| dx dy + \mu \int_{\Omega} (g_1^2 + g_2^2) dx dy.$$

subject to the constraint

$$f - u - g_{1,x} - g_{2,y} = 0. \quad (\text{A.1.2})$$

The Euler-Lagrange equations for this functional can be calculated by including the constraint with a Lagrange multiplier λ^{con} (not to be confused with λ in Equa-

tion A.1.1). The obtained system is

$$K(u) = -\lambda^{con} \quad (\text{A.1.3})$$

$$2\mu g_1 = -\lambda_x^{con} = \frac{\partial}{\partial x} K(u) \quad (\text{A.1.4})$$

$$2\mu g_2 = -\lambda_y^{con} = \frac{\partial}{\partial y} K(u), \quad (\text{A.1.5})$$

where $K(u) = \text{div}\left(\frac{\nabla u}{|\nabla u|}\right)$ is the curvature of the level lines of u . Back-substituting g_1 and g_2 into the constraint $f - u - g_{1,x} - g_{2,y} = 0$ yields

$$f - u = \frac{1}{2\mu} \left(\frac{\partial^2}{\partial x^2} K(u) + \frac{\partial^2}{\partial y^2} K(u) \right) = \frac{1}{2\mu} \Delta K(u). \quad (\text{A.1.6})$$

As stated in Chapter 5, the evolution equation for u in the model of Osher, Solé and Vese is:

$$u_t = -\frac{1}{2\lambda} \Delta \left[\text{div} \left(\frac{\nabla u}{|\nabla u|} \right) \right] - (u - f), \quad (\text{A.1.7})$$

and since at equilibrium $u_t = 0$, the two Equations A.1.6 and A.1.7 are the same (μ and λ can be identified as being the same parameters upon comparing the two models). So, for the modified Vese-Osher model, u can be evolved according to Equation A.1.7, and at any iteration, g_1 and g_2 can be solved for using Equations A.1.4 and A.1.5 respectively, without having to resort to solving for them at each iteration with separate PDEs. The previous derivation is significant because it is much simpler than that in [4], not requiring the Hodge decomposition or proof of the equivalence of gradient descent and Laplacian of gradient ascent evolution to arrive at the ultimate solution.

An example of the subcomponents obtained from running this modified Vese-

Osher model is shown in Figure A.1. Observe that the g_1 subcomponent contains most of the horizontal energy/edges of the image, while the g_2 subcomponent contains most of the vertical energy and edges. These could be used for discrimination, as done in Chapters 3 and 4. However there is one potential difficulty - the slow speed of the gradient descent solution of the model, due to the requirement of a small time step.

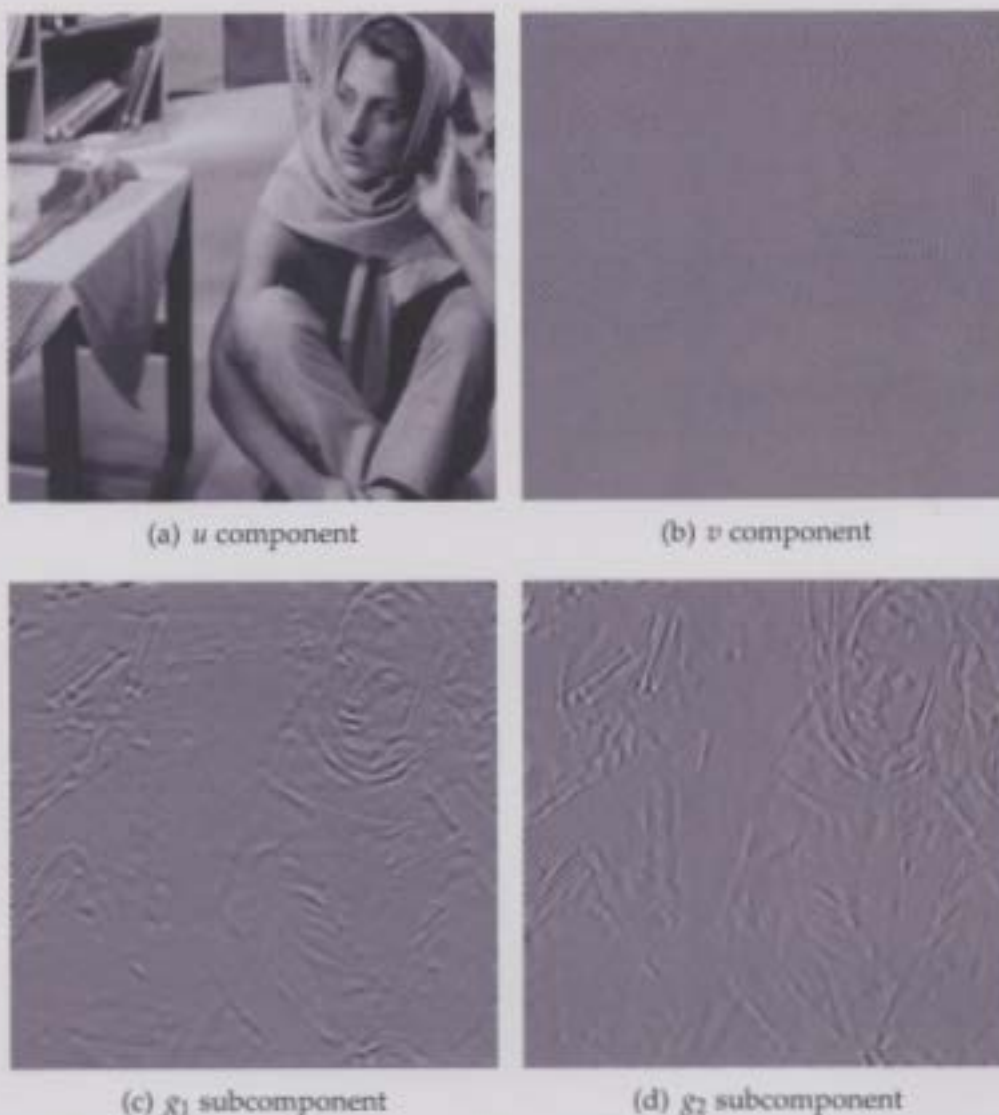


Figure A.1: Osher-Solé-Vese decomposition of barbara with analytical g_1 and g_2 components after 314 iterations

Difference of Convex Functionals

Solution of IES

B.1 Difference of Convex Functionals Solution

B.1.1 Some Elementary Convex Analysis

Some basic convex analysis is reviewed in this subsection, mostly adapted from [103]. Recall that a function h is called convex on Ω if

$$h(tx + (1 - t)y) \leq th(x) + (1 - t)h(y) \quad (\text{B.1.1})$$

for all $t \in [0, 1]$ and $x, y \in \Omega$. An example of a convex function on the positive reals is shown in Figure B.1.

If a function $h \in C^2(\Omega)$, the class of continuously differentiable functions up to second order on Ω , then it is convex if and only if $h'' \geq 0$ on Ω . It is clear that

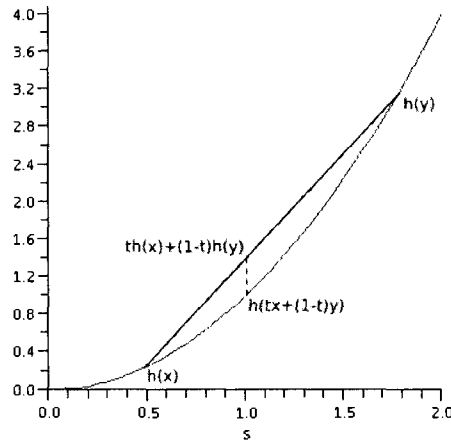


Figure B.1: Example of convex function h

the integral of a convex function is also convex. Also, if a functional is convex in a function, then it is also convex in the partial derivatives of the function, because of linearity of the partial differentiation operation.

An alternate definition of a convex function is one such that its tangent plane at any point lies below the graph of the function. However, this is not defined for non-differentiable functions. Thus, a generalization of the gradient for a convex function can be made. This is called the subgradient. The subgradient of a convex function h is defined to be any vector s such that

$$f(z) \geq f(x) + \langle s, z - x \rangle \quad \forall z.$$

Here $\langle c, d \rangle$ denotes the inner or dot product of c and d . In general, there will be more than one subgradient, and the set of all subgradients of a convex function h at a point x is called the subdifferential of h at x , written as $\partial h(x)$.

One observation that makes an iterative solution possible for IES is that its energy functional can be expressed as the difference of two convex functionals in each of u , g_1 and g_2 . This method for the provably convergent iterative solution of IES is based in the next subsection.

B.1.2 Difference of Convex Functionals Solution of IES

Recall the entire form of the energy functional $\mathbf{E}_{\text{IES}}(u, g_1, g_2)$ for IES (an abbreviated form where the Vese-Osher terms were not expanded was given in Equation 3.5.1)

$$\begin{aligned} \mathbf{E}_{\text{IES}}(u, g_1, g_2) = & \underbrace{\int_{\Omega} |\nabla u| dx dy}_{\text{Term 1}} + \underbrace{\lambda \int_{\Omega} (f - u - \partial_x g_1 - \partial_y g_2)^2 dx dy}_{\text{Term 2}} + \\ & \underbrace{\mu \left[\int_{\Omega} \sqrt{g_1^2 + g_2^2} dx dy \right]}_{\text{Term 3}} - \underbrace{\gamma_1 \int_{\Omega} \frac{|\nabla u|}{G_{\sigma} * |\nabla \vec{g}| + \epsilon_1} dx dy}_{\text{Term 4}} - \underbrace{\gamma_2 \int_{\Omega} \frac{G_{\sigma} * |\nabla \vec{g}|}{|\nabla u| + \epsilon_2} dx dy}_{\text{Term 5}}. \end{aligned} \quad (\text{B.1.2})$$

Clearly, the term $\lambda \int_{\Omega} (f - u - g_{1,x} - g_{2,y})^2 dx dy$ is convex in u because the second derivative with respect to u is $2 \geq 0$. The difference of convex functional solution framework has the following three steps, assuming that the image space is isomorphic to $\mathbb{R}^{m \times n}$ and the difference of convex (d.c.) functional has the form $F = G - H$, where G and H are convex functionals:

1. Choose $x^{(0)}$ in $\mathbb{R}^{m \times n}$
2. Set $y^{(k)}$ in $\partial H(x^{(k)})$

3. Set $x^{(k+1)}$ in $\partial G^*(y^{(k)})$.

Above, the G^* refers to the convex conjugate of the functional G , defined as:

$$G^*(y) = \sup\{\langle y, x \rangle - f(x) | x \in \Omega\}, \quad (\text{B.1.3})$$

once again $\langle y, x \rangle$ denoting the inner product of the two vectors x and y .

By Corollary 23.5.1 from [104], the subdifferential of the conjugate of a function is equal to the inverse subdifferential of the function itself. In other words, $x^{(k+1)} = \partial G^*(y^{(k)}) = (\partial G)^{-1}(y^{(k)})$. Therefore, the above three-step procedure can be condensed to initialization followed by iteration of the equation $\partial G(x^{(k+1)}) = \partial H(x^{(k)})$, since $y^{(k)}$ is equal to both. Also, because only continuous functionals are considered, the subdifferential ∂ corresponds to the usual gradient ∇ . For functionals, which are present instead of functions for this variational problem, it is easy to show that the gradient corresponds to the first variation of the functional. So, for example for u , the equation $G_u(u^{(k+1)}) = H_u(u^{(k)})$ is obtained, where G and H are both convex functionals, and e.g. $G_u(u) = \frac{\partial G}{\partial u} - \frac{d}{dx} \frac{\partial G}{\partial u_x} - \frac{d}{dy} \frac{\partial G}{\partial u_y}$. Table B.1 shows which terms are convex in u , g_1 and g_2 , which can then be used to find the function-dependent G and H functionals, from which the iterative equations for the cartoon component u and the texture subcomponents g_1 and g_2 can be derived.

Function	Terms included in G	Terms included in H
u	1, 2, 5	4
g_1	2, 3, 4	5
g_2	2, 3, 4	5

Table B.1: Terms in IES functional contained in each convex functional in DC functional

Thus, the following implicit equation to find u at timestep $k + 1$ from that at timestep k is derived

$$\begin{aligned}
& -\operatorname{div} \left(\frac{\nabla u^{(k+1)}}{|\nabla u^{(k+1)}|} \right) + 2\lambda(u^{(k+1)} + g_{1,x}^{(k)} + g_{2,y}^{(k)} - f) \\
& -\operatorname{div} \left(\frac{\gamma_2(G_\sigma * |\nabla \vec{g}^{(k)}|) \nabla u^{(k+1)}}{|\nabla u^{(k+1)}|^3} \right) = -\operatorname{div} \left(\frac{\gamma_1 \nabla u^{(k)}}{(G_\sigma * |\nabla \vec{g}^{(k)}|) |\nabla u^{(k)}|^3} \right). \quad (\text{B.1.4})
\end{aligned}$$

Similarly, for g_1 and g_2 , the following implicit iterative equations are found

$$\begin{aligned}
& 2\lambda(f_x - u_x^{(k+1)} - g_{1,xx}^{(k+1)} - g_{2,xy}^{(k)}) + \mu \frac{g_1^{(k+1)}}{\sqrt{(g_1^{(k+1)})^2 + (g_2^{(k)})^2}} \\
& -\operatorname{div} \left(\frac{\gamma_1 |\nabla u^{(k+1)}| (G_\sigma * \nabla g_1^{(k+1)})}{(G_\sigma * |\nabla \vec{g}^{(k+1)}|)^3} \right) = -\operatorname{div} \left(\frac{\gamma_2 (G_\sigma * \nabla g_1^{(k)})}{|\nabla u| G_\sigma * |\nabla \vec{g}^{(k)}|} \right) \quad (\text{B.1.5})
\end{aligned}$$

$$\begin{aligned}
& 2\lambda(f_y - u_y^{(k+1)} - g_{1,xy}^{(k+1)} - g_{2,yy}^{(k+1)}) + \mu \frac{g_2^{(k+1)}}{\sqrt{(g_1^{(k+1)})^2 + (g_2^{(k+1)})^2}} \\
& -\operatorname{div} \left(\frac{\gamma_1 |\nabla u^{(k+1)}| (G_\sigma * \nabla g_2^{(k+1)})}{(G_\sigma * |\nabla \vec{g}^{(k+1)}|)^3} \right) = -\operatorname{div} \left(\frac{\gamma_2 (G_\sigma * \nabla g_2^{(k)})}{|\nabla u| G_\sigma * |\nabla \vec{g}^{(k)}|} \right). \quad (\text{B.1.6})
\end{aligned}$$

These equations are all highly non-linear, so they can't be solved directly. What makes them especially problematic is the future values of the functions at timestep $k + 1$ in the denominators of some of the divergence terms. As an approximation however, these future values in the denominators can be changed to current values

at timestep k . This changes the iterative equations to the following

$$\begin{aligned}
& -\operatorname{div} \left(\frac{\nabla u^{(k+1)}}{|\nabla u^{(k)}|} \right) + 2\lambda(u^{(k+1)} + g_{1,x}^{(k)} + g_{2,y}^{(k)} - f) \\
& -\operatorname{div} \left(\frac{\gamma_2(G_\sigma * |\nabla \vec{g}^{(k)}|) \nabla u^{(k+1)}}{|\nabla u^{(k)}|^3} \right) = -\operatorname{div} \left(\frac{\gamma_1 \nabla u^{(k)}}{(G_\sigma * |\nabla \vec{g}^{(k)}|) |\nabla u^{(k)}|^3} \right) \quad (\text{B.1.7})
\end{aligned}$$

$$\begin{aligned}
& 2\lambda(f_x - u_x^{(k+1)} - g_{1,xx}^{(k+1)} - g_{2,xy}^{(k)}) + \mu \frac{g_1^{(k+1)}}{\sqrt{(g_1^{(k)})^2 + (g_2^{(k)})^2}} \\
& -\operatorname{div} \left(\frac{\gamma_1 |\nabla u^{(k+1)}| (G_\sigma * \nabla g_1^{(k+1)})}{(G_\sigma * |\nabla \vec{g}^{(k)}|)^3} \right) = -\operatorname{div} \left(\frac{\gamma_2 (G_\sigma * \nabla g_1^{(k)})}{|\nabla u^{(k+1)}| G_\sigma * |\nabla \vec{g}^{(k)}|} \right) \quad (\text{B.1.8})
\end{aligned}$$

$$\begin{aligned}
& 2\lambda(f_y - u_y^{(k+1)} - g_{1,xy}^{(k+1)} - g_{2,yy}^{(k+1)}) + \mu \frac{g_2^{(k+1)}}{\sqrt{(g_1^{(k+1)})^2 + (g_2^{(k)})^2}} \\
& -\operatorname{div} \left(\frac{\gamma_1 |\nabla u^{(k+1)}| (G_\sigma * \nabla g_2^{(k+1)})}{(G_\sigma * |\nabla \vec{g}^{(k)}|)^3} \right) = -\operatorname{div} \left(\frac{\gamma_2 (G_\sigma * \nabla g_2^{(k)})}{|\nabla u^{(k+1)}| G_\sigma * |\nabla \vec{g}^{(k)}|} \right). \quad (\text{B.1.9})
\end{aligned}$$

Then, Additive Operator Splitting can be used to solve these equations at each iteration because each of them is semi-implicit, and basically of the form $h_t = A(h^{(k)})h^{(k+1)}$, where A is a suitably defined matrix, derived from Equations B.1.7-B.1.9 above, and primarily of the nonlinear diffusion form $h_t = \operatorname{div}(d(|\nabla h|^2) \nabla h)$, where d is a diffusivity function. A Difference of Convex functionals solution of Improved Edge Segregation was not implemented in this thesis, however, this discussion is valuable in that it proves that there is a method to solve the IES equations that converges to the correct solution.

Bibliography

- [1] Y. Meyer, *Oscillating Patterns in Image Processing and Nonlinear Evolution Equations: The Fifteenth Dean Jacqueline B. Lewis Memorial Lectures*. American Mathematical Society, 2001. iii, 1, 6, 7, 11, 39, 40, 41, 59, 181, 182, 184
- [2] L. A. Vese and S. J. Osher, "Modeling textures with total variation minimization and oscillating patterns in image processing," *Journal of Scientific Computing*, vol. 19, December 2003. iii, xxii, xxiii, 8, 11, 15, 42, 43, 47, 48, 59, 60, 66, 68, 83, 87, 89, 96, 99, 101, 107, 112, 113, 123, 127, 181, 182, 188
- [3] T. Chan and L. Vese, "An active contour model without edges," *Scale Space Theories in Computer Vision, Lecture Notes in Computer Science*, vol. 1682, pp. 141–151, February 2001. iii, 47, 50, 51, 53, 112, 113, 127
- [4] S. J. Osher, A. Solé, and L. A. Vese, "Image decomposition and restoration using total variation minimization and the H^{-1} norm," *Multiscale Modeling and Simulation: A SIAM Interdisciplinary Journal*, vol. 1, no. 3, pp. 349–370, 2003. iii, 11, 12, 15, 17, 45, 46, 64, 86, 87, 88, 94, 140, 142, 144, 145, 154, 155, 157, 181, 247

- [5] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 7, pp. 629–639, 1990. iv, 5, 31, 32, 155, 156, 157, 208
- [6] <http://vision.ece.ucsb.edu/segmentation/edgeflow>, "Vision research lab - a multiresolution approach to image segmentation based on edgeflow." xxi, 2
- [7] A. N. Venetsanaoupoulos and K. N. Plataniotis, *Color Image Processing and Applications*. Springer, 2000. 3
- [8] <http://vision.ece.ucsb.edu/segmentation/edgeflow>, "Vision research lab - a multiresolution approach to image segmentation." 3
- [9] W. Y. Ma and B. S. Manjunath, "Edge flow: a framework for boundary detection and image segmentation," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), San Juan, Puerto Rico*, pp. 744–749, June 1997. 3
- [10] R. M. Haralick, "Statistical and structural approaches to texture," *Proceedings of the IEEE*, vol. 67, pp. 786–804, May 1979. 3
- [11] J. Frank, *Three-Dimensional Electron Microscopy of Macromolecular Assemblies: Visualization of Biological Molecules in their Native State*. Oxford University Press, USA, 2006. 3
- [12] R. Garnett, T. Huegerich, C. Chui, and W. He, "A universal noise removal algorithm with an impulse detector," *IEEE Transactions on Image Processing*, vol. 14, no. 11, pp. 1747–1754, Nov. 2005. 3

- [13] M. Nikolova, "A variational approach to remove outliers and impulse noise," *Journal of Mathematical Imaging and Vision*, vol. 20, no. 1,2, pp. 99–120, Jan. 2004. 4
- [14] <http://www.geoeye.com/gallery/ioweeek/archive/02-07-22/index.htm>, "GeoEye image of the week: July 22, 2002." xxi, 4
- [15] <http://www.sprawls.org/ppmi2/>, "The physical principles of medical imaging." 4
- [16] <http://ai.stanford.edu/~ruzon/NASA/texture.html>, "Texture segmentation on mars." xxi, 5, 6
- [17] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 674–693, July 1989. 6
- [18] D. Marr, *Vision*. W. H. Freeman and Company, 1982. 9, 64
- [19] G. Sharma, *Digital Color Imaging Handbook*. CRC Press, 2003. 11
- [20] R. Carter, *Mapping the Mind*. Orion Books Ltd., 2004. 11
- [21] N. Sprljan and E. Izquierdo, "New perspectives on image compression using a cartoon-texture decomposition model," in *4th EURASIP Conference on Video/Image Processing and Multimedia Communications, Zagreb, Croatia*, pp. 359–368, July 2003. 11
- [22] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proceedings of SIGGRAPH 2000, New Orleans, USA, July 2000*. 11, 197

- [23] D. Paquin, D. Levy, E. Schreibmann, and L. Xing, "Multiscale image registration," *Mathematical Biosciences and Engineering*, vol. 3, pp. 389–418, 2006. 11, 240
- [24] L. Bar, N. Sochen, and N. Kiryati, "Variational pairing of image segmentation and blind restoration," in *Proc. 8th European Conference on Computer Vision (ECCV 2004), Prague, Czech Republic, Part II: Lecture Notes in Computer Science no. 3022*, pp. 166–177, Springer, May 2004. 13, 111
- [25] R. Shahidi and C. Moloney, "Simultaneous textured image decomposition and discrimination preserving large-scale structure," in *Proceedings of the International Symposium on Signal Processing and Information Technology, Vancouver, BC, August 2006*. 13, 111, 114, 116
- [26] V. Caselles and J. Morel, "Introduction to the special issue on partial differential equations and geometry-driven diffusion in image processing and analysis," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 269–273, 1998. 20
- [27] J. J. Koenderink, "The structure of images," *Biol. Cyber.*, vol. 50, pp. 363–370, 1984. 20, 26
- [28] A. P. Witkin, "Scale-space filtering," in *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1019–1021, 1983. 20
- [29] G. Aubert and P. Kornprobst, *Mathematical Problems in Image Processing - Partial Differential Equations and the Calculus of Variations*. Springer-Verlag, 2002. 23, 25, 50

- [30] D. Kincaid and W. Cheney, *Numerical Analysis: Mathematics of Scientific Computing*, 3rd. Edition. Brooks/Cole, 2002. 25, 28
- [31] B. Jähne, *Digital Image Processing*, 5th edition. Springer-Verlag, 2002. 27, 54, 157
- [32] J. Weickert, "A Review of nonlinear diffusion filtering," in *Scale-Space Theory in Computer Vision 1997*, pp. 3–28, Springer, 1997. 31, 32
- [33] M. Giaquinta and S. Hildebrandt, *Calculus of Variations I: The Lagrangian Formalism*. Springer, 1996. 34
- [34] J. W. Thomas, *Numerical Methods for Partial Differential Equations*. Springer, 1999. 34
- [35] F. Catte, P. L. Lions, J. M. Morel, and T. Coll, "Image selective smoothing and edge detection by nonlinear diffusion," *SIAM J. Numer. Anal.*, vol. 29, no. 1, pp. 182–193, 1992. 35
- [36] J. Weickert, B. M. ter Haar Romeny, and M. A. Viergever, "Efficient and reliable schemes for nonlinear diffusion filtering," *IEEE Transactions on Image Processing*, vol. 7, pp. 398–410, March 1998. 36, 38, 73, 92, 121, 238
- [37] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd Edition. The Johns Hopkins University Press, 1996. 36
- [38] L. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Phys. D.*, vol. 60, pp. 259–268, 1992. 39, 40, 92, 217, 218

- [39] J.-F. Aujol and A. Chambolle, "Dual norms and image decomposition models," Tech. Rep. 5130, INRIA Technical Report, March 2004. 39, 181
- [40] I. Daubechies and G. Teschke, "Variational image restoration by means of wavelets: simultaneous decomposition, deblurring and denoising," *Applied and Computational Harmonic Analysis*, 2004. 39, 94, 132
- [41] F. Malgouyres, "Minimizing the total variation under a general convex constraint for image restoration," *IEEE Transactions on Image Processing*, vol. 11, no. 12, pp. 1450–1456, 2002. 40
- [42] T. F. Chan and C. K. Wong, "Total variation blind deconvolution," *IEEE Transactions on Image Processing*, vol. 7, pp. 370–375, March 1998. 40
- [43] W. Yin, D. Goldfarb, and S. Osher, "A comparison of three total variation based texture extraction models," *J. Vis. Comun. Image Represent.*, vol. 18, no. 3, pp. 240–252, 2007. 41
- [44] W. Yin, D. Goldfarb, and S. Osher, "Image cartoon-texture decomposition and feature selection using the total variation regularized L^1 functional," in *Variational, Geometric, and Level Set Methods in Computer Vision, Lecture Notes in Computer Science 3752*, pp. 73–84, Springer, 2005. 42
- [45] G. Aubert and L. Vese, "A variational method in image recovery," *SIAM J. Numer. Anal.*, vol. 34, no. 5, pp. 1948–1979, 1997. 43, 127
- [46] B. Julesz and J. R. Bergen, "Textons, the fundamental elements in preattentive vision and perception of textures," *The Bell System Technical Journal*, vol. 62, pp. 1619–1645, July-August 1983. 47

- [47] S. Osher and J. Sethian, "Fronts propagating with curvature dependent speed: algorithms based on the Hamilton-Jacobi formulation," *Journal of Computational Physics*, vol. 79, pp. 12–49, 1988. 48
- [48] G. Sapiro, *Geometric Partial Differential Equations and Image Analysis*. Cambridge University Press, 2001. 48, 50
- [49] C. L. Epstein and M. Gage, *The curve shortening flow*. Springer-Verlag, 1987. 50
- [50] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," *International Journal of Computer Vision*, vol. 22, no. 1, pp. 61–77, 1997. 50
- [51] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes - active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1987. 50
- [52] L. A. Vese and T. F. Chan, "A Multiphase level set framework for image segmentation using the Mumford-Shah model," *International Journal of Computer Vision*, vol. 50, no. 3, pp. 271–293, 2002. 51
- [53] T. F. Chan, B. Y. Sandberg, and L. A. Vese, "Active contours without edges for vector-valued images," *Journal of Visual Communication and Image Representation*, vol. 11, pp. 130–141, June 2000. 51, 122
- [54] T. Brox, J. Weickert, B. Burgeth, and P. Mrázek, "Nonlinear structure tensors," *Image and Vision Computing*, vol. 24, pp. 41–55, Jan. 2006. 55, 188
- [55] J. Weickert, "Coherence-enhancing diffusion filtering," *Int. J. Comput. Vision*, vol. 31, no. 2-3, pp. 111–127, 1999. 56, 162, 164, 207

- [56] R. Terebes, O. Laviaille, P. Baylou, and M. Borda, "Mixed anisotropic diffusion," *ICPR*, vol. 3, pp. 760–763, 2002. 56, 165
- [57] H. Dörrie, *100 Great Problems of Elementary Mathematics*. Dover Publications, 1965. 62
- [58] J.-F. Aujol and T. F. Chan, "Combining geometrical and textured information to perform image classification," Tech. Rep. 04-65, CAM, UCLA, November 2004. 67
- [59] K. A. Patwardhan and G. Sapiro, "Automatic image decomposition," in *ICIP*, pp. 645–648, 2004. 70, 89
- [60] J. Rittscher and J. Sullivan, "An integral criterion for detecting boundary edges and textured regions," in *Proceedings of the 15th International Conference on Pattern Recognition, Barcelona, Spain*, vol. 3, pp. 1064–1067, 2000. 75
- [61] <http://www-unix.mcs.anl.gov/mpi/mpich/>, "MPICH2 home page." 77, 105
- [62] <http://restoreinpaint.sourceforge.net/>, "Image restoration." 77
- [63] <http://www.libpng.org/pub/png/libpng.html>, "libpng Home Page." 77
- [64] M. Snir and S. Otto and S. Huss-Lederman and D. Walker and J. Dongarra, *MPI - The Complete Reference, Volume 1, the MPI Core*. The MIT Press, 1998. 77
- [65] P. Brodatz, *Textures: A Photographic Album for Artists and Designers*. Dover Publications, Inc., New York, 1966. 80

- [66] <http://sipi.usc.edu/database/database.cgi?volume=textures>, "USC-SIPI Image Database - Textures." 80
- [67] <http://dynamo.ecn.purdue.edu/~eigenman/ECE563/Handouts/parallelPDEsolvers.ppt>, "Parallel solver for partial differential equations." xxii, 82, 83
- [68] R. Shahidi and C. Moloney, "Variational textured image decomposition with improved edge segregation," in *Proceedings of the International Conference on Image Processing, Atlanta, GA, Oct. 2006*. 94, 148, 181
- [69] <http://www.clustermatic.org>, "Clustermatic: a complete cluster solution." 104
- [70] <http://www-unix.mcs.anl.gov/perfvis/download/index.htm>, "Performance visualization for parallel programs." 105
- [71] A. C. Bovik, M. Clark, and W. S. Geisler, "Multichannel texture analysis using localized spatial filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 55–73, January 1990. 107
- [72] C. Samson, L. Blanc-Feraud, J. Zerubia, and G. Aubert, "Simultaneous image classification and restoration using a variational approach," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 618–623, 1999. 111
- [73] X. Bresson and J. Thiran, "Image segmentation model using active contour and image decomposition," in *Proceedings of the IEEE International Conference on Image Processing 2006, Atlanta, USA, 2006*. 112, 207, 243

- [74] T. L. Saaty and P. C. Kainen, *The Four-Color Problem: Assaults and Conquest*. Dover Publications, Inc., New York, 1986. 112
- [75] R. Kimmel and A. M. Bruckstein, "Regularized laplacian zero crossings as optimal edge integrators," *Int. J. Comput. Vision*, vol. 53, no. 3, pp. 225–243, 2003. 120
- [76] C. Li, C. Xi, C. Gui, and M. D. Fox, "Level set evolution without reinitialization: a new variational formulation," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 120
- [77] T. Brox and J. Weickert, "A TV flow based local scale estimate and its application to texture discrimination," Tech. Rep. 134, Department of Mathematics, Saarland University, Saarbrücken, Germany, April 2005. 121, 122, 138
- [78] J.-F. Aujol and G. Gilboa, "Constrained and snr-based solutions for tv-hilbert space image denoising," *J. Math. Imaging Vis.*, vol. 26, no. 1-2, pp. 217–237, 2006. 140, 155
- [79] D. Tschumperlé, *PDE's Based Regularization of Multivalued Images and Applications*. PhD thesis, Université de Nice - Sophia Antipolis, 2002. 162, 163, 179, 198
- [80] T. McGraw, B. Vemuri, Y. Chen, M. Rao, and T. Mareci, "Dtmri denoising and neuronal fiber tracking," *Med. Image Anal.*, vol. 8, no. 2, pp. 95–111, Jun. 2004. 180

- [81] A. A. Altun and N. Allahverdi, "Recognition of fingerprints enhanced by contourlet transform with artificial neural networks," in *Proc. of 28th Int. Conf. Information Technology Interfaces 2006*, 167-172, June 2006. 180
- [82] R. Shahidi and C. Moloney, "Textured image decomposition and discrimination with improved edge segregation," in *Proceedings of the IEEE NECEC 2005*, November 2005. 181
- [83] R. Shahidi and C. Moloney, "Orientation-adaptive decomposition," in *Proceedings of the IEEE NECEC, St. John's, Canada*, November 2006. 181, 204
- [84] R. Shahidi and C. Moloney, "An orientation-adaptive image decomposition model and its application to denoising," in *Proceedings of the International Workshop on Nonlinear Signal and Image Processing 2007, Bucharest, Romania*, September 2007. 181, 206
- [85] H. Farid and E. P. Simoncelli, "Differentiation of multi-dimensional signals," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 496–508, 2004. 189, 190
- [86] J. Xiao, H. Cheng, H. Sawhney, C. Rao, and M. Isnardi, "Bilateral filtering-based optical flow estimation with occlusion detection," in *Proceedings of the European Conference on Computer Vision, Graz, Austria*, May 2006. 198
- [87] <http://www.maplesoft.com>, "Math software for engineers, educators & students | maplesoft." 199
- [88] <http://www.phasevision.co.uk>, "Phase vision homepage." 202

- [89] <http://www.stanford.edu/group/radar/snaphu/>, "SNAPHU website." 202
- [90] C. W. Chen and H. A. Zebker, "Network approaches to two-dimensional phase unwrapping: intractability and two new algorithms," *Journal of the Optical Society of America A*, vol. 17, pp. 401–414, 2000. 202
- [91] J. Buckland, J. Huntley, and S. Turner, "Unwrapping noisy phase maps by use of a minimum-cost-matching algorithm," *Appl. Opt.*, vol. 34, no. 23, pp. 5100–5108, 1995. 202
- [92] P. Charbonnier, G. Aubert, M. Blanc-Féraud, and M. Barlaud, "Two deterministic half-quadratic regularization algorithms for computed imaging," in *Proceedings of the International Conference on Image Processing (ICIP), Austin, Texas, USA*, vol. 2, pp. 168–172, 1994. 208
- [93] A. R. Bruss, "The eikonal equation: some results applicable to computer vision," *Journal of Mathematical Physics*, vol. 23, no. 5, pp. 890–896, 1982. 209
- [94] J. Sethian, "A fast marching level set method for monotonically advancing fronts," in *Proc. Nat. Acad. Sci.*, vol. 93, pp. 1591–1595, 1996. 211
- [95] <http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=6110>, "MATLAB central file exchange - toolbox fast marching." 211
- [96] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the Fourth Alvey Vision Conference, Manchester*, pp. 147–151, 1988. 212

- [97] J. Immwekaer, "Fast noise variance estimation," *Comput. Vis. Image Underst.*, vol. 64, no. 2, pp. 300–302, 1996. 216, 242
- [98] T. Aach, C. Mota, I. Stuke, M. Muhlich, and E. Barth, "Analysis of superimposed oriented patterns," *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3690–3700, 2006. 242
- [99] P. Blomgren and T. F. Chan, "Color TV: total variation methods for restoration of vector-valued images," *IEEE Transactions on Image Processing*, vol. 7, pp. 304–309, March 1998. 244
- [100] J.-F. Aujol and S. H. Kang, "Color image decomposition and restoration," *Journal of Visual Communication and Image Registration*, vol. 17, August 2006. 244
- [101] L. A. Vese and S. J. Osher, "Color texture modeling and color image decomposition in a variational-pde approach," *synasc*, vol. 0, pp. 103–110, 2006. 244
- [102] I. Galic, J. Weickert, M. Welk, A. Bruhn, A. Belyaev, and H.-P. Seidel, "Towards pde-based image compression," *Variational, Geometric and Level Set Methods in Computer Vision, Lecture Notes in Computer Science*, vol. 3752, pp. 37–48, 2005. 244
- [103] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms I*. Springer, 1993. 249
- [104] R. T. Rockafellar, *Convex Analysis*. Princeton University Press, 1970. 252

